Basic SQL Interview Q&As (Part 1)

🔹 Section 1: Fundamentals

Q1. What is SQL, and why is it used?
Answer: SQL (Structured Query Language) is the standard language for working with relational databases. It allows users to **create, read, update, and delete (CRUD)** data stored in tables. SQL is widely used because it provides a powerful, declarative way to interact with large datasets, making data management simple and efficient.
Example:

```
SELECT * FROM employees;
```

💡 **Tip:** Always mention — "SQL is the universal language for RDBMS like MySQL, PostgreSQL, Oracle."

Q2. What are the main types of SQL dialects? Give examples.
Answer: SQL dialects are vendor-specific implementations of SQL. Although all follow the SQL standard, each adds unique features.

- **MySQL** → open-source, widely used for web apps
- **PostgreSQL** → advanced features, strong ACID compliance
- **SQLite** → lightweight, file-based, used in mobile apps
- **Oracle SQL** → enterprise-focused, powerful security & scalability
- **T-SQL** (SQL Server) → Microsoft's SQL extension
  💡 **Tip:** If asked, always name the dialect you've **actually worked with**.

Q3. What is the difference between SQL and NoSQL databases?
Answer:

- **SQL databases** are relational, use predefined schemas, and store data in structured tables. They follow ACID properties, making them ideal for financial, transactional, and enterprise systems.

- **NoSQL databases** are non-relational, schema-less, and handle unstructured/semi-structured data. They provide flexibility and scale horizontally, commonly used in big data and real-time apps (e.g., MongoDB, Cassandra).
  💡 **Tip:** Mention that SQL = structured, consistent data; NoSQL = flexible, scalable systems.

Q4. What is a database and why do we need it?
Answer: A database is an organized collection of structured information stored electronically. It helps in efficient **storage, retrieval, management, and security** of data. Databases are essential because manual file-based systems are inefficient and prone to redundancy and inconsistency.
💡 **Tip:** Relating this to real-world systems (e.g., online shopping cart, student records) makes your answer stronger.

Q5. What is DBMS? What are its types?
Answer: A Database Management System (DBMS) is software that manages data in databases. It provides tools to define, manipulate, and secure data.

- **Relational DBMS (RDBMS):** stores data in tables (MySQL, Oracle, PostgreSQL)
- **Hierarchical DBMS:** organizes data in tree structures (IBM IMS)
- **Network DBMS:** data represented as records connected by links
- **Object-oriented DBMS:** handles complex objects (db4o)
- **Graph DBMS:** focuses on relationships (Neo4j)
  💡 **Tip:** Interviewers often expect you to highlight **RDBMS** first.

Q6. What is RDBMS? Give examples.
Answer: RDBMS (Relational Database Management System) is the most common type of DBMS. It organizes data into **related tables** linked by keys (primary and foreign). SQL is the language used to interact with RDBMS. Examples include MySQL, PostgreSQL, Oracle, and MariaDB.

Q7. What is a table, row, and column in SQL?
Answer:

- **Table**: collection of related data stored in rows and columns.
- **Row (Record/Tuple)**: a single entry in a table (e.g., one student's details).
- **Column (Field/Attribute)**: a vertical structure representing data type (e.g., Name, Age).

Q8. What is an SQL statement? Give examples.
Answer: SQL statements are commands executed by the SQL engine to perform specific tasks. Examples:

- SELECT → retrieve data
- INSERT → add records
- DELETE → remove records
- CREATE → create a new table
  💡 **Tip:** Mention that statements fall under categories like DDL, DML, DQL, DCL, TCL.

Q9. What are the different types of SQL commands?
Answer:

- **DDL (Data Definition Language):** defines structure → CREATE, DROP, ALTER
- **DML (Data Manipulation Language):** manipulates data → INSERT, UPDATE, DELETE
- **DCL (Data Control Language):** controls access → GRANT, REVOKE
- **TCL (Transaction Control Language):** manages transactions → COMMIT, ROLLBACK
- **DQL (Data Query Language):** queries data → SELECT

Q10. Give examples of SQL commands for each type.

- **DDL:**

```
CREATE TABLE students (id INT, name VARCHAR(50));
```

- **DML:**

INSERT INTO students VALUES (1,'Aman');

- **DCL:**

GRANT SELECT ON students TO user1;

- **TCL:**

COMMIT;

- **DQL:**

SELECT * FROM students;

🔹 Section 2: Querying Data

Q11. What is an SQL query? What are select and action queries?
Answer: An SQL query is a request to the database to perform a task.

- **Select queries**: used to fetch data **(SELECT * FROM employees;)**
- **Action queries**: modify data **(INSERT, UPDATE, DELETE).**
  💡 **Tip:** Always explain with one simple select + one action example.

Q12. How do you select all columns from a table?

SELECT * FROM employees;

Q13. How do you select specific columns from a table?

SELECT name, salary FROM employees;

Q14. What is the DISTINCT keyword and why is it used?
Answer: DISTINCT eliminates duplicate rows and returns only unique values from a column. Useful for analyzing unique categories.

SELECT DISTINCT department FROM employees;

💡 **Tip:** Mention that DISTINCT helps clarity but adds overhead on large datasets.

Q15. How do you filter rows using WHERE?

```sql
SELECT * FROM employees WHERE salary > 50000;
```

Q16. What are comparison, logical, and set operators in SQL?

- **Comparison:** =, !=, <, >
- **Logical:** AND, OR, NOT
- **Set:** IN, BETWEEN, EXISTS

Q17. How do you use BETWEEN, IN, and LIKE for filtering?

```sql
SELECT * FROM students WHERE age BETWEEN 18 AND 22;
```

```sql
SELECT * FROM students WHERE city IN ('Delhi','Mumbai');
```

```sql
SELECT * FROM employees WHERE name LIKE 'A%';
```

💡 **Tip:** LIKE with % and _ is very frequently asked in interviews.

% is used for **flexible pattern searches** (start, end, contains).

_ is used for **strict length-based matches** (character-by-character).

Q18. What is a NULL value? How is it different from zero or blank?
Answer:

- **NULL** = unknown/missing value
- **Zero** = numeric value
- **Blank (‘ ’)** = empty string of length zero
  💡 **Tip:** NULL is **not equal** to anything, even another NULL.
- `SELECT * FROM students WHERE city = NULL; -- No results`
- `SELECT * FROM students WHERE city IS NULL; --  Correct`
- Always stress the difference:
- **NULL** → unknown / missing
- **0** → number
- **''** → empty text

Q19. How do you sort records using ORDER BY?

```sql
SELECT * FROM employees ORDER BY salary DESC;
```

Q20. How do you limit the number of rows returned (LIMIT/TOP)?

- **MySQL/PostgreSQL:**

```
SELECT * FROM employees LIMIT 5;
```

- **SQL Server:**

```
SELECT TOP 5 * FROM employees;
```

🔹 Section 3: Aggregate Functions & Grouping

Q21. What are aggregate functions in SQL? Give examples.
Answer: Aggregate functions perform calculations on a set of values and return a single result. Common examples:

- COUNT() → counts rows
- SUM() → total of a column
- AVG() → average value
- MIN() & MAX() → lowest and highest values

📌 Example Table: **employees**

emp_id	name	department	salary
1	Aman	IT	60000
2	Riya	HR	40000
3	Kabir	IT	65000
4	Neha	Finance	55000
5	Arjun	HR	42000

1. **COUNT()** → counts rows

```
SELECT COUNT(*) AS total_employees

FROM employees;
```

Output:

total_employees
5

💡 COUNT(column) ignores NULL values, while COUNT(*) counts all rows.

2. **SUM()** → total of a column

```
SELECT SUM(salary) AS total_salary
FROM employees;
```

Output:

total_salary
262000

3. **AVG()** → average value

```
SELECT AVG(salary) AS avg_salary
FROM employees;
```

Output:

avg_salary
52400

4. **MIN() & MAX()** → lowest & highest values

```
SELECT MIN(salary) AS lowest, MAX(salary) AS highest
FROM employees;
```

Output:

lowest	highest
40000	65000

5. **Aggregate with GROUP BY**

```
SELECT department, COUNT(*) AS emp_count, AVG(salary) AS avg_salary
FROM employees
GROUP BY department;
```

Output:

department	emp_count	avg_salary
IT	2	62500
HR	2	41000
Finance	1	55000

🔑 Quick Cheat Sheet

Function	Purpose	Example	Output
COUNT()	Count rows	COUNT(*)	5
SUM()	Total values	SUM(salary)	262000
AVG()	Average values	AVG(salary)	52400
MIN()	Lowest value	MIN(salary)	40000
MAX()	Highest value	MAX(salary)	65000

Q22. How do you count rows in a table?

```
SELECT COUNT(*) FROM employees;
```

Q23. How do you calculate sum, average, minimum, and maximum?

```
SELECT SUM(salary), AVG(salary), MIN(salary), MAX(salary)

FROM employees;
```

Q24. What is GROUP BY and why is it used?
Answer: Groups rows that have the same values in specified columns, often used with aggregate functions.

```
SELECT department, AVG(salary)

FROM employees

GROUP BY department;
```

Q25. What is HAVING and how is it different from WHERE?
Answer:

- WHERE filters rows before grouping.
- HAVING filters groups after aggregation.

```
SELECT department, COUNT(*)
```

```
FROM employees
GROUP BY department
HAVING COUNT(*) > 5;
```

Q26. How do you find the nth highest or lowest value in a column?

Answer: You can find the **nth highest/lowest value** using:

1. ORDER BY + LIMIT + OFFSET (MySQL, PostgreSQL, SQLite).
2. ROW_NUMBER() or RANK() window functions (SQL Server, Oracle, PostgreSQL).
3. Nested subqueries.

1. Using LIMIT + OFFSET (MySQL, PostgreSQL, SQLite)

Find the 3rd highest salary:

```
SELECT salary
FROM employees
ORDER BY salary DESC
LIMIT 1 OFFSET 2;
```

Output:

salary
60000

💡 **OFFSET 2 skips the top 2 rows, then LIMIT 1 fetches the 3rd row.**

2. Using Subquery (Works in most SQL dialects)

Find the 2nd highest salary:

```
SELECT MAX(salary)
```

```
FROM employees
WHERE salary < (SELECT MAX(salary) FROM employees);
```

Output:

max
65000

💡 **You can nest further to get 3rd, 4th highest, but it gets messy.**

3. Using ROW_NUMBER() (SQL Server, Oracle, PostgreSQL)

Find the 3rd highest salary:

```
SELECT salary
FROM (
  SELECT salary, ROW_NUMBER() OVER (ORDER BY salary DESC) AS row_num
  FROM employees
) ranked
WHERE row_num = 3;
```

Output:

salary
60000

4. Using RANK() (handles duplicates properly)

```
SELECT salary
FROM (
  SELECT salary, RANK() OVER (ORDER BY salary DESC) AS rank_num
  FROM employees
) ranked
WHERE rank_num = 3;
```

- If two employees share the same salary, RANK() will give them the same rank.
- ROW_NUMBER() does not allow ties (each row gets a unique number).

.