Day 52 – Basic SQL Interview Q&As (Part 2)

🔹 Section 4: Joins & Relationships

Q27. What is a JOIN in SQL, and why is it used?
Answer: A JOIN combines rows from two or more tables based on a related column (usually keys). It helps retrieve meaningful data spread across multiple tables.

```
SELECT employees.name, departments.dept_name
FROM employees
INNER JOIN departments
ON employees.dept_id = departments.id;
```

Output:

name	dept_name
Rahul	HR
Ananya	IT

💡 *Tip:* Interviewers often ask you to explain JOINs with real-world examples like “employees and departments.”

Q28. What are the types of JOINs in SQL?

- **INNER JOIN** → returns only matching rows.
- **LEFT JOIN** → all rows from left + matched rows from right.
- **RIGHT JOIN** → all rows from right + matched rows from left.
- **FULL JOIN** → all rows from both sides.
- **CROSS JOIN** → Cartesian product (all combinations).
- **SELF JOIN** → table joined with itself.

Example: LEFT JOIN

```
SELECT e.name, d.dept_name
FROM employees e
LEFT JOIN departments d
ON e.dept_id = d.id;
```

💡 *Tip:* Always say INNER JOIN is the most common in interviews.

Q29. What is a PRIMARY KEY and why is it important?
Answer: A PRIMARY KEY uniquely identifies each record in a table. It combines **NOT NULL** + **UNIQUE** constraints.

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

💡 *Tip:* Each table should ideally have one primary key.

Q30. What is a UNIQUE key, and how is it different from PRIMARY KEY?

- **PRIMARY KEY** → Only one per table, cannot be NULL.
- **UNIQUE KEY** → Multiple allowed per table, allows one NULL.

```
CREATE TABLE users (
 id INT PRIMARY KEY,
 email VARCHAR(100) UNIQUE
);
```

💡 *Tip:* Say: *“Primary key identifies rows, unique key ensures uniqueness of values.”*

Q31. What is a FOREIGN KEY and why is it used?
Answer: A FOREIGN KEY links one table to another by referencing the

PRIMARY KEY of another table. Ensures **referential integrity**.

```
CREATE TABLE orders (
  order_id INT PRIMARY KEY,
  user_id INT,
  FOREIGN KEY (user_id) REFERENCES users(id)
);
```

💡 *Tip:* Interviewers may ask: “What happens if you try to insert a record with a non-existent foreign key?” (It fails).

Q32. What are database relationships?

- **One-to-One** → one record relates to one record (passport ↔ person).
- **One-to-Many** → one record relates to many (department ↔ employees).
- **Many-to-Many** → many records relate to many (students ↔ courses).

💡 *Tip:* Always give real-life examples (students & courses is classic).

🔹 Section 5: Data Manipulation

Q33. How do you insert data into a table?

```
INSERT INTO students (id, name, age)
VALUES (1, 'Aman', 20);
```

💡 *Tip:* For multiple rows, use INSERT INTO ... VALUES (...), (...);

Q34. How do you update data in a table?

```
UPDATE students
```

```sql
SET age = 21
WHERE id = 1;
```

💡 *Tip:* Without WHERE, it updates all rows — common mistake.

Q35. How do you delete records from a table?

```sql
DELETE FROM students WHERE id = 1;
```

💡 *Tip:* DELETE removes selected rows, TRUNCATE removes all rows.

Q36. How do you add, rename, or delete a column in SQL?

```sql
ALTER TABLE students ADD email VARCHAR(100);
ALTER TABLE students RENAME COLUMN email TO student_email;
ALTER TABLE students DROP COLUMN student_email;
```

💡 *Tip:* ALTER is often asked in interviews.

Q37. How do you create or drop a table?

```sql
CREATE TABLE teachers (
  id INT PRIMARY KEY,
  name VARCHAR(50)
);

DROP TABLE teachers;
```

💡 *Tip:* DROP removes structure + data.

🔹 Section 6: Indexes, Views & Constraints

Q38. What is an index, and why is it important?
Answer: Index is a data structure that improves query performance

by reducing scan time.

```
CREATE INDEX idx_name ON employees(name);
```

💡 *Tip:* Index speeds up SELECT but slows INSERT/UPDATE.

Q39. What are clustered and non-clustered indexes?

- **Clustered** → rearranges actual table rows (only one per table).
- **Non-clustered** → separate structure storing references (can be many).

💡 *Tip:* Always mention: *Primary key = clustered index by default in many DBs.*

Q40. What is a VIEW and why use it?
Answer: A VIEW is a virtual table created from a query.

```
CREATE VIEW high_salary_employees AS
SELECT name, salary FROM employees WHERE salary > 60000;
```

💡 *Tip:* Say: *Views simplify queries & restrict data access.*

Q41. What are SQL constraints and why are they important?
Constraints enforce rules on data.

- **PRIMARY KEY, FOREIGN KEY, UNIQUE, NOT NULL, DEFAULT, CHECK**.

💡 *Tip:* They ensure **data integrity**.

🔷 Section 7: Subqueries & Aliases

Q42. What is a subquery?
Answer: A query inside another query.

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

💡 *Tip:* Mention correlated vs non-correlated subqueries.

Q43. What is an alias in SQL?

```
SELECT name AS employee_name FROM employees;
```

```
SELECT e.name FROM employees e;
```

💡 *Tip:* Aliases make queries short & readable.

🔹 Section 8: Data Types & Operators

Q44. What are Boolean and numeric data types?

- Boolean → TRUE/FALSE/NULL.
- Numeric → INT, FLOAT, DECIMAL.

Q45. What operators are used in SQL?

- Arithmetic (+, -, *, /)
- Comparison (=, >, <, !=)
- Logical (AND, OR, NOT)
- String (LIKE, %, _)
- Set (UNION, INTERSECT, MINUS)

Q46. How do you select even or odd records?

```
SELECT * FROM students WHERE id % 2 = 0; -- Even
```

```
SELECT * FROM students WHERE id % 2 <> 0; -- Odd
```

🔹 Section 9: Advanced Basics

Q47. What is normalization, and why is it used?

Answer:

Normalization is the process of organizing data in a database to **reduce redundancy and improve data integrity**.

- **1NF (First Normal Form):** Ensure atomic (indivisible) values, no repeating groups.
- **2NF (Second Normal Form):** Be in 1NF + remove partial dependency (non-key attributes depend on whole primary key).
- **3NF (Third Normal Form):** Be in 2NF + remove transitive dependency (non-key attributes depend only on the primary key).

💡 **Tip:** In interviews, mention up to **3NF** (commonly asked). Higher forms (BCNF, 4NF, 5NF) are less frequent in basics.

Q48. What is denormalization, and when is it used?
Answer: Denormalization introduces redundancy for faster reads.

💡 *Tip:* Used in reporting systems.

Q49. What is the difference between DELETE, TRUNCATE, and DROP?

- **DELETE** → removes rows (DML, reversible with ROLLBACK).
- **TRUNCATE** → removes all rows (DDL, cannot rollback).
- **DROP** → deletes table + structure.

Q50. What is the difference between LEFT JOIN and LEFT OUTER JOIN?
Answer: No difference. OUTER is optional.

Q51. What is the difference between renaming a column and aliasing it?

- **Rename** → permanent schema change.
- **Alias** → temporary for query readability.

Q52. What is the CASE statement in SQL?

```
SELECT name,
    CASE
        WHEN salary > 60000 THEN 'High'
        ELSE 'Low'
    END AS salary_category
FROM employees;
```

💡 *Tip:* Say CASE = if-else in SQL.

Q53. What is the difference between SQL and PL/SQL?

- **SQL** → data manipulation language.
- **PL/SQL** → procedural extension with loops, conditions, etc.

Q54. What are common SQL challenges in projects?

- Query optimization.
- Indexing strategy.
- Handling NULLs.
- Ensuring data integrity.
- Managing concurrent transactions.

small bonus:

Q55. Can a table have multiple foreign keys?

Answer:→ Yes.

Q56. Can we create a view from another view?

Answer: Yes, but avoid nesting too much.

Q57. Can we still use a view if base table is deleted?

Answer:No, view becomes invalid.