SQL Interview Questions & Answers (First 40)

1. What is SQL and why is it used?

SQL (Structured Query Language) is a standard language used to interact with relational databases. It is used to store, retrieve, update, delete, and manage data efficiently.

Example:
SELECT * FROM employee;

2. Types of joins available in SQL

Joins are used to combine data from multiple tables based on a related column.

Types:
- INNER JOIN
- LEFT JOIN
- RIGHT JOIN
- FULL OUTER JOIN
- CROSS JOIN
- SELF JOIN
- SEMI JOIN
- ANTI JOIN

Example:
SELECT * FROM emp e INNER JOIN dept d ON e.dept_id = d.dept_id;

3. Difference between Primary Key and Foreign Key

Primary Key uniquely identifies each record in a table and cannot contain NULL values.
Foreign Key establishes a relationship between two tables and can contain NULL values.

Example:
PRIMARY KEY(emp_id)
FOREIGN KEY(dept_id) REFERENCES department(dept_id);

4. What is a Composite Key?

A composite key is a primary key made up of two or more columns to uniquely identify a row.

Example:
PRIMARY KEY(order_id, product_id);

5. Difference between Inner Join and Outer Join

INNER JOIN returns only matching records from both tables. OUTER JOIN returns matching records along with non-matching records.

Example:
SELECT * FROM A LEFT JOIN B ON A.id = B.id;

6. Normalization and Denormalization

Normalization is the process of organizing data to reduce redundancy and improve data integrity. Denormalization combines tables to improve query performance, mainly used in reporting systems.

7. What is a Subquery?

A subquery is a query nested inside another query. It is used for filtering, comparisons, and calculations.

Example:
SELECT * FROM employee WHERE salary > (SELECT AVG(salary) FROM employee);

8. What is CTE and why is it used?

CTE (Common Table Expression) is a temporary result set defined using WITH clause. It improves readability and simplifies complex queries.

Example:
WITH emp_cte AS (SELECT * FROM employee) SELECT * FROM emp_cte;

9. Difference between Table and View

A table stores data physically in the database. A view is a virtual table created using a SQL query and does not store data.

Example:
CREATE VIEW emp_view AS SELECT emp_id, name FROM employee;

10. DML and DQL Statements

DML statements are used to manipulate data, while DQL is used to retrieve data.

DML: INSERT, UPDATE, DELETE
DQL: SELECT

Example:
INSERT INTO emp VALUES (1, 'Ram', 5000);

11. DDL Statements

DDL statements are used to define or modify database structures.

Statements: CREATE, ALTER, DROP, TRUNCATE

Example:
CREATE TABLE emp (id INT, name VARCHAR(20));

12. TCL and DCL Statements

TCL manages transactions and DCL controls access to data.

TCL: COMMIT, ROLLBACK, SAVEPOINT
DCL: GRANT, REVOKE

Example:
COMMIT;

13. Window Functions

Window functions perform calculations across a set of rows related to the current row without grouping.

Examples:
ROW_NUMBER(), RANK(), DENSE_RANK(), LEAD(), LAG()

Example:
SELECT salary, RANK() OVER(ORDER BY salary DESC) FROM emp;

14. Difference between RANK and DENSE_RANK

RANK assigns same rank for duplicate values and skips the next rank. DENSE_RANK does not skip rank numbers.

Example:
SELECT RANK() OVER(ORDER BY salary DESC), DENSE_RANK() OVER(ORDER BY salary DESC) FROM emp;

15. LEAD and LAG Functions

LAG is used to access previous row data and LEAD is used to access next row data.

Example:
SELECT salary, LAG(salary) OVER(ORDER BY emp_id) FROM emp;

16. Handling Duplicate and NULL Values

Duplicates can be identified using GROUP BY and HAVING clause. NULL values can be handled using IFNULL or COALESCE.

Example:
SELECT name FROM emp GROUP BY name HAVING COUNT(*) > 1;

17. Difference between OLTP and OLAP

OLTP systems handle day-to-day transactions and are highly normalized. OLAP systems are used for analytical reporting and are denormalized.

18. SQL vs NoSQL

SQL databases use structured schema and ACID properties. NoSQL databases support flexible schema and are used for big data applications.

Examples: MySQL (SQL), MongoDB (NoSQL)

19. Database vs Data Warehouse

A database stores current operational data. A data warehouse stores historical data for reporting and analysis.

20. What is Stored Procedure?

A stored procedure is a precompiled SQL code stored in the database and reused multiple times.

Example:
CREATE PROCEDURE get_emp() BEGIN SELECT * FROM emp; END;

21. ACID Properties

ACID ensures reliable database transactions.
A - Atomicity
C - Consistency
I - Isolation
D - Durability

22. CAP Theorem

CAP theorem states that a distributed system can guarantee only two of the three: Consistency, Availability, and Partition Tolerance.

23. Fact Table vs Dimension Table

Fact tables store numerical transactional data. Dimension tables store descriptive attributes.

Example: Sales (Fact), Customer (Dimension)

24. Types of Slowly Changing Dimensions

SCD Type 0 (No change)
SCD Type 1 (Overwrite)
SCD Type 2 (History)
SCD Type 3 (Limited history)

25. Inner Join vs Outer Join Example

Inner Join returns only matching records. Outer Join includes unmatched records.

Example:
SELECT * FROM A INNER JOIN B ON A.id=B.id;

26. Left Anti Join

Left Anti Join returns records from left table that do not exist in right table.

Example:
SELECT * FROM A LEFT JOIN B ON A.id=B.id WHERE B.id IS NULL;

27. Left Semi Join

Left Semi Join returns records from left table that have matching records in right table.

Example:
SELECT * FROM A WHERE EXISTS (SELECT 1 FROM B WHERE A.id=B.id);

28. Incremental Data Handling

Incremental load processes only new or updated records using timestamp or ID.

Example:
WHERE updated_at > (SELECT MAX(updated_at) FROM target);

29. Difference between TRUNCATE and DELETE

DELETE removes rows one by one and supports rollback. TRUNCATE removes all rows and cannot be rolled back.

30. Recursive Stored Procedure

A recursive stored procedure is a procedure that calls itself until a condition is met.

31. CHAR vs VARCHAR

CHAR stores fixed-length strings. VARCHAR stores variable-length strings and saves space.

32. Print Duplicate Values

Duplicate values can be identified using GROUP BY and HAVING clause.

Example:
SELECT col, COUNT(*) FROM table GROUP BY col HAVING COUNT(*)>1;

33. Remove Duplicates using ID

Duplicates can be removed using self join.

Example:
DELETE t1 FROM emp t1 JOIN emp t2 ON t1.name=t2.name AND t1.id>t2.id;

34. Star Schema vs Snowflake Schema

Star schema is simple and fast. Snowflake schema is normalized and complex.

35. Hash Distribution Table

Hash distribution evenly distributes data across nodes using a hash key.

Used in MPP databases.

36. Types of SCD Tables

SCD Type 0, Type 1, Type 2, Type 3.

37. Implement SCD Type 2

SCD Type 2 maintains history by adding start_date, end_date, and is_current columns.

38. DROP vs TRUNCATE vs DELETE

DROP removes the table structure. TRUNCATE removes all records. DELETE removes selected records.

39. ETL vs ELT

ETL transforms data before loading. ELT loads raw data first and transforms later.

40. 3rd Highest Salary Query

Use window function to find nth highest salary.

Example:
SELECT * FROM (SELECT salary, ROW_NUMBER() OVER(ORDER BY salary DESC) rnk FROM emp) t WHERE rnk=3;

SQL PRACTICAL QUESTIONS – EMPLOYEE & BONUS TABLE

Create Employee and Bonus Tables

```
CREATE TABLE employee (

   emp_id INT PRIMARY KEY,

   emp_name VARCHAR(50),

   salary INT,

   dept_id INT,

   manager_id INT

);


CREATE TABLE bonus (

   emp_id INT,

   bonus INT

);
```

---

1. Find 3rd Highest Salary – Normal SQL

Explanation:
Uses nested subqueries to eliminate highest and second highest salaries.

```
SELECT MAX(salary)

FROM employee

WHERE salary < (

   SELECT MAX(salary)

   FROM employee

   WHERE salary < (SELECT MAX(salary) FROM employee)

);
```

---

2. Find 3rd Highest Salary – Window Function

Explanation:
ROW_NUMBER assigns ranking based on salary.

```
SELECT emp_id, emp_name, salary

FROM (

   SELECT *, ROW_NUMBER() OVER (ORDER BY salary DESC) AS rnk

   FROM employee

) t

WHERE rnk = 3;
```

---

3. Find 3rd Highest Salary – CTE

Explanation:
CTE improves readability and reusability.

```
WITH ranked_emp AS (

   SELECT *, ROW_NUMBER() OVER (ORDER BY salary DESC) AS rnk

   FROM employee

)

SELECT emp_id, emp_name, salary

FROM ranked_emp

WHERE rnk = 3;
```

---

4. Stored Procedure – Highest Salary Department Wise

```
DELIMITER //

CREATE PROCEDURE dept_highest_salary()

BEGIN

   SELECT dept_id, MAX(salary) AS max_salary
   FROM employee
   GROUP BY dept_id;
END //
DELIMITER ;

CALL dept_highest_salary();
```

---

5. Count Employees per Department

```
SELECT dept_id, COUNT(*) AS emp_count
FROM employee
GROUP BY dept_id;
```

---

6. Create Test_Employees Table and Insert Records

```
CREATE TABLE test_employees LIKE employee;

INSERT INTO test_employees
SELECT * FROM employee LIMIT 10;
```

---

7. LEAD and LAG Example

```
SELECT emp_id, salary,
     LAG(salary) OVER (ORDER BY emp_id) AS prev_salary,
     LEAD(salary) OVER (ORDER BY emp_id) AS next_salary
FROM employee;
```

---

8. Even and Odd Employee IDs

```
-- Even
```

```
SELECT * FROM employee WHERE emp_id % 2 = 0;


-- Odd

SELECT * FROM employee WHERE emp_id % 2 <> 0;
```

---

9. Employee Salary Greater than Manager Salary

```
SELECT e.emp_name, e.salary, m.emp_name AS manager_name

FROM employee e

JOIN employee m ON e.manager_id = m.emp_id

WHERE e.salary > m.salary;
```

---

10. Sum of Salary per Department

```
SELECT dept_id, SUM(salary) AS total_salary

FROM employee

GROUP BY dept_id;
```

---

11. Employee Name and Manager Name in Same Table

```
SELECT e.emp_name AS employee,

    m.emp_name AS manager

FROM employee e

LEFT JOIN employee m

ON e.manager_id = m.emp_id;
```

---

12. Duplicate Employee Names

```
SELECT emp_name, COUNT(*)

FROM employee
```

```
GROUP BY emp_name

HAVING COUNT(*) > 1;
```

---

13. Employees Without Salary

```
SELECT e.emp_id, e.emp_name

FROM employee e

LEFT JOIN bonus b ON e.emp_id = b.emp_id

WHERE b.salary IS NULL;
```

---

14. Print Duplicate Records in a Table

```
SELECT column_name, COUNT(*)

FROM table_name

GROUP BY column_name

HAVING COUNT(*) > 1;
```

---

15. Duplicate Records for All Columns

```
SELECT emp_id, emp_name, salary, dept_id, manager_id, COUNT(*)

FROM employee

GROUP BY emp_id, emp_name, salary, dept_id, manager_id

HAVING COUNT(*) > 1;
```

---

16. Non-Matching Records from Tables A and B

```
SELECT A.*

FROM A

LEFT JOIN B ON A.id = B.id

WHERE B.id IS NULL;
```

---

17. Remove Vowels and Spaces from String

```
SELECT REGEXP_REPLACE(
    'I aM VeRy StRoN At CoDiNg',
    '[AEIOUaeiou ]',
    ''
) AS result;
```

---

18. Join Behavior with NULL Values

Explanation:
INNER JOIN ignores NULL values.

```
SELECT *
FROM A
INNER JOIN B ON A.col = B.col;
```

---

19. Hash Distribution Table

```
CREATE TABLE emp_hash (
    emp_id INT,
    emp_name VARCHAR(50)
)
DISTRIBUTED BY HASH(emp_id);
```

---

20. Print Duplicate Values with Count

```
SELECT emp_id, COUNT(*) AS cnt
FROM emp
GROUP BY emp_id
```

```
HAVING COUNT(*) > 1;
```

---

21. Total Revenue by Product Category

```
SELECT c.category_name,
    SUM(s.sale_amount) AS total_revenue
FROM sales s
JOIN categories c
ON s.category_id = c.category_id
GROUP BY c.category_name;
```

---

22. Join Result Counts (Inner, Left, Right, Outer, Left Anti)

```
-- Inner Join
SELECT COUNT(*)
FROM table1 t1
INNER JOIN table2 t2 ON t1.col = t2.col;

-- Left Join
SELECT COUNT(*)
FROM table1 t1
LEFT JOIN table2 t2 ON t1.col = t2.col;

-- Right Join
SELECT COUNT(*)
FROM table1 t1
RIGHT JOIN table2 t2 ON t1.col = t2.col;
```

```
-- Full Outer Join
SELECT COUNT(*) FROM (
  SELECT t1.col FROM table1 t1
  LEFT JOIN table2 t2 ON t1.col = t2.col
  UNION
  SELECT t2.col FROM table1 t1
  RIGHT JOIN table2 t2 ON t1.col = t2.col
) x;


-- Left Anti Join
SELECT COUNT(*)
FROM table1 t1
LEFT JOIN table2 t2 ON t1.col = t2.col
WHERE t2.col IS NULL;
```

Git Interview Questions & Answers

---

1. What is Git?

Git is a **distributed version control system** used to track changes in source code.
It allows multiple developers to work together and manage code history efficiently.

---

2. What is Version Control?

Version control is a system that keeps track of file changes over time.
It helps in restoring previous versions and managing collaboration.

---

3. Difference between Git and GitHub

Git is a **version control tool** used locally.
GitHub is a **cloud-based platform** that hosts Git repositories and supports collaboration.

---

4. What is a Repository?

A repository is a storage location where Git tracks all project files and their versions.
It can be **local** or **remote**.

---

5. What is a Commit?

A commit is a snapshot of changes saved to the repository with a message.

```
git commit -m "Added login feature"
```

---

6. What is a Branch?

A branch allows you to work on features independently without affecting main code.

```
git branch feature_login
```

---

7. What is Merge?

Merge combines changes from one branch into another branch.

```
git merge feature_login
```

---

8. What is Rebase?

Rebase reapplies commits on top of another branch to create a linear history.

```
git rebase main
```

---

9. Difference between Merge and Rebase

- **Merge** keeps full commit history
- **Rebase** rewrites commit history

---

10. What is Clone?

Clone creates a local copy of a remote repository.

```
git clone https://github.com/user/repo.git
```

---

11. What is Pull?

Pull fetches and merges changes from a remote repository.

```
git pull origin main
```

---

12. What is Push?

Push uploads local commits to a remote repository.

```
git push origin main
```

---

13. What is Staging Area?

The staging area holds changes before they are committed.

```
git add file.txt
```

---

14. What is git status?

Displays the current state of the working directory and staging area.

git status

---

15. What is git log?

Shows the commit history of the repository.

git log

---

16. What is git diff?

Shows differences between commits, branches, or files.

git diff

---

17. What is git reset?

Used to undo commits or unstage files.

git reset HEAD~1

---

18. What is git revert?

Creates a new commit that reverses a previous commit.

git revert commit_id

---

19. Difference between git reset and git revert

- **Reset** removes commits from history
- **Revert** keeps history intact

---

20. What is Conflict in Git?

A conflict occurs when Git cannot automatically merge changes from different branches.

---

21. How do you resolve merge conflicts?

Edit the conflicting files manually, then add and commit them.

```
git add .

git commit -m "Resolved conflict"
```

---

22. What is .gitignore?

Specifies files and folders that Git should ignore.

```
*.log

.env

node_modules/
```

---

23. What is HEAD in Git?

HEAD points to the current branch and the latest commit.

---

24. What is Fork?

A fork is a copy of a repository created under your own GitHub account.

---

25. What is Cherry-pick?

Cherry-pick applies a specific commit from one branch to another.

```
git cherry-pick commit_id
```

---

26. What is Tag?

A tag marks a specific commit, usually for releases.

```
git tag v1.0
```

---

27. What is git stash?

Temporarily saves uncommitted changes.

```
git stash
```

---

28. What is git fetch?

Downloads changes from remote without merging.

```
git fetch origin
```

---

29. Difference between git fetch and git pull

- **Fetch** downloads changes only
- **Pull** downloads and merges changes

---

30. What is a Bare Repository?

A bare repository has no working directory and is used for sharing code.

---

31. What is git blame?

Shows who modified each line of a file.

```
git blame file.txt
```

---

32. What is git reflog?

Tracks all changes made to HEAD and branches.

```
git reflog
```

---

33. What is Squash?

Squash combines multiple commits into a single commit.

---

34. What is Pull Request?

A pull request is a request to merge changes after code review.

---

35. What is Continuous Integration with Git?

CI automatically builds and tests code whenever changes are pushed.

---

36. What is Monorepo?

A monorepo stores multiple projects in a single repository.

---

37. What is Distributed Version Control?

Each developer has a full copy of the repository and its history.

---

38. What is git config?

Used to configure Git settings.

```
git config --global user.name "John"
```

---

39. What is git checkout?

Used to switch branches or restore files.

```
git checkout main
```

---

40. Git Workflow

A common workflow uses **main**, **feature branches**, and **pull requests** for controlled development.