Top 60 DBMS Interview Questions with Answers for 2025

Last Updated : 28 Apr, 2025

A **Database Management System (DBMS)** is the backbone of modern data storage and management. Understanding DBMS concepts is critical for anyone looking to work with databases. Whether you're preparing for your first job in database management or advancing in your career, being well-prepared for a DBMS interview can make all the difference.

In this article, we've covered a list of **DBMS interview questions** that cover everything from basic to advanced topics. These questions will help us build a solid understanding of DBMS concepts, from how databases are structured to complex query optimization. Start your preparation today to secure your dream role in database management!

Basic DBMS Interview Questions

DBMS Basic Interview Questions are designed to test your foundational understanding of database management systems. These questions focus on key concepts such as the differences between relational and non-relational databases, basic DBMS operations, data types, and database structure. A strong grasp of these basics will help you confidently approach more advanced topics.

1. What is a Database Management System (DBMS)?

A **Database Management System (DBMS)** is software that is used to manage and organize databases. It provides an interface to interact

security, and managing concurrency. Examples include MySQL, PostgreSQL, Oracle, and SQL Server.

2. What are the advantages of using a DBMS?

The advantages of using a DBMS are:

- **Data Integrity:** Ensures that the data is accurate and consistent.
- **Data Security:** Provides controlled access to sensitive data by setting permissions for different users.
- **Efficient Data Retrieval:** Optimizes queries and indexing, allowing faster data retrieval.
- **Reduced Redundancy:** Avoids duplicate data by enforcing normalization.
- **Backup and Recovery:** Offers automatic backup and recovery mechanisms.
- **Concurrent Access:** Allows multiple users to access the database at the same time without conflicts.

3. What is the difference between DBMS and RDBMS?

- **DBMS (Database Management System):** A system that allows users to create, store, modify, and delete data. It does not require a relational structure for data organization. Examples: Microsoft Access, XML databases.
- **RDBMS (Relational Database Management System):** A subset of DBMS that stores data in a structured format, using tables (relations), and supports relational operations like joins. It enforces data integrity through keys and supports SQL for querying. Examples: MySQL, Oracle, SQL Server.

4. What are the different types of DBMS?

- **Network DBMS:** Data is represented as a graph with many-to-many relationships. Example: Integrated Data Store (IDS).
- **Relational DBMS (RDBMS):** Data is organized in tables (relations) and managed through SQL. Example: MySQL, PostgreSQL.
- **Object-Oriented DBMS:** Data is stored as objects, like in object-oriented programming. Example: ObjectDB.

5. What is a relation in DBMS?

A **relation** in DBMS is a table that consists of rows and columns. Each row represents a record, and each column represents an attribute or property of the entity being described. Relations are defined by a schema, which specifies the attributes (columns) of the table.

6. What is a table in DBMS?

A **table** in DBMS is a collection of data organized in rows and columns. It is the primary structure for storing data in a relational database. Each row represents an entity (record), and each column represents an attribute of that entity.

7. What are rows and columns in a DBMS?

- **Rows (Tuples):** A row represents a single record or entity. Each row contains values for each attribute (column).
- **Columns (Attributes):** A column represents a property or characteristic of the entity. Each column has a data type, such as integer, string, etc

8. What are the primary components of a DBMS?

The primary components of a DBMS are:

- **Database Engine:** Responsible for storing, retrieving, and managing

- **Query Processor:** Interprets and executes SQL queries.
- **Transaction Manager:** Ensures the ACID properties of transactions.
- **Storage Manager:** Manages the physical storage of data.

9. What is a primary key? Explain with an example.

A **Primary Key** is a unique identifier for each record in a table. It ensures that no two records have the same value for the primary key field. It cannot contain **NULL** values. **Example:** In a `STUDENT` table, `ROLL_NO` could be the primary key because each student has a unique roll number.

ROLL_NO	NAME	ADDRESS
1	Ram	Delhi
2	Suresh	Delhi

10. What is a foreign key? Explain with an example.

A **Foreign Key** is an attribute in a table that links to the primary key in another table. It creates a relationship between two tables, ensuring referential integrity. **Example:** In a `STUDENT` table, the `BRANCH_CODE` could be a foreign key referencing the primary key `BRANCH_CODE` in the `BRANCH` table.

Student Table

ROLL_NO	NAME	BRANCH_CODE
1	Ram	CS
2	Suresh	IT

BRANCH_CODE	BRANCH_NAME
CS	Computer Science
IT	Information Technology

Here, `BRANCH_CODE` in `STUDENT` is a foreign key referencing `BRANCH_CODE` in `BRANCH`.

11. What is normalization? Why is it important in DBMS?

Normalization is the process of organizing data in a way that reduces redundancy and dependency. It involves dividing large tables into smaller ones and defining relationships between them to ensure data integrity.

Importance:

- Eliminates redundant data.
- Prevents anomalies during data operations (insertion, update, deletion).
- Improves data integrity and consistency.

12. What is denormalization? How does it differ from normalization?

Denormalization is the process of combining tables to improve query performance, often by introducing redundancy. While normalization minimizes redundancy, denormalization sacrifices some of it to improve speed for read-heavy operations.

Difference:

- **Normalization:** Focuses on minimizing redundancy and improving data integrity.

13. What is a candidate key in DBMS?

A **Candidate Key** is a set of one or more attributes that can uniquely identify a tuple in a relation. A relation can have multiple candidate keys, and one of them is chosen as the primary key.

14. What is the use of the SQL SELECT statement?

The **SELECT** statement is used to query data from one or more tables. It allows you to retrieve specific columns or all columns, optionally applying filters (WHERE), sorting (ORDER BY), and joining multiple tables.

Example:

```
SELECT NAME, AGE FROM STUDENT WHERE AGE > 18;
```

15. What is a view in DBMS? How does it differ from a table?

A **View** is a virtual table created by querying one or more base tables. It does not store data physically but dynamically retrieves it when queried. Unlike a table, a view does not store its own data but presents data from other tables.

16. What are the different types of relationships in DBMS?

The three main types of relationships in DBMS are:

- **One-to-One (1:1):** A record in one table is associated with a single record in another table.
- **One-to-Many (1:M):** A record in one table is associated with multiple records in another table.
- **Many-to-Many (M:M):** Multiple records in one table are associated with multiple records in another table.

A **schema** in DBMS is the structure that defines the organization of data in a database. It includes tables, views, relationships, and other elements. A schema defines the tables and their columns, along with the constraints, keys, and relationships.

18. What are constraints in DBMS?

Constraints in DBMS are rules that limit the type of data that can be inserted into a table to ensure data integrity and consistency. Common types of constraints include: **NOT NULL, PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK, DEFAULT.**

19. What is the difference between DELETE and TRUNCATE in SQL?

- **DELETE:** Deletes specific rows from a table based on a condition. It logs each row deletion and can be rolled back.
- **TRUNCATE:** Removes all rows from a table without logging individual row deletions. It cannot be rolled back and is faster than DELETE.

20. What is an index in DBMS and how is it used?

An **index** is a data structure that improves the speed of data retrieval operations on a database table. It works like a table of contents in a book, allowing the database to quickly find the location of a record based on a column value.

21. What is the role of the Database Administrator (DBA)?

A **Database Administrator (DBA)** is responsible for managing and overseeing the entire database environment. Their key responsibilities include:

- **Backup and Recovery:** Ensuring regular backups and providing recovery solutions in case of failures.
- **Performance Tuning:** Monitoring and optimizing the database's performance.
- **Security Management:** Managing user access, privileges, and enforcing security policies.
- **Data Integrity:** Ensuring data consistency and integrity through constraints and checks.
- **Upgrades and Patches:** Keeping the database software up-to-date with patches and upgrades.
- **Troubleshooting:** Identifying and resolving database-related issues.

22. What is an entity-relationship diagram (ERD)?

An **Entity-Relationship Diagram (ERD)** is a visual representation of the entities within a system and the relationships between those entities. It is used in database design to model the structure of data and how different pieces of data relate to each other.

- **Entities**: Objects or things within the system (e.g., `Student`, `Course`).
- **Attributes**: Properties or details about an entity (e.g., `Student Name`, `Course Duration`).
- **Relationships**: How entities interact with each other (e.g., `Student` enrolls in `Course`).

Example of ERD:

- A `Student` entity might have attributes like `ID`, `Name`, and `Age`.
- A `Course` entity might have attributes like `CourseID`, `CourseName`.
- A relationship `Enrolls` connects `Student` and `Course` with attributes like `EnrollmentDate`.

23. What is a join in SQL? Name and explain different types of joins.

to query data from multiple tables in a relational database.

Here are the different types of **joins** in SQL:

1. **INNER JOIN:** Returns only the rows where there is a match in both tables.

Example:

```
SELECT * FROM Student
INNER JOIN Course ON Student.ID = Course.StudentID;
```

2. **LEFT JOIN (or LEFT OUTER JOIN)**: Returns all rows from the left table and the matching rows from the right table. If there is no match, NULL values will be returned for columns from the right table.

Example:

```
SELECT * FROM Student
LEFT JOIN Course ON Student.ID = Course.StudentID;
```

3. RIGHT JOIN (or RIGHT OUTER JOIN): Returns all rows from the right table and the matching rows from the left table. If there is no match, NULL values will be returned for columns from the left table.

Example:

```
SELECT * FROM Student
RIGHT JOIN Course ON Student.ID = Course.StudentID;
```

4. FULL JOIN (or FULL OUTER JOIN): Returns all rows when there is a match in either the left or the right table. If there is no match, NULL values will be returned for the columns of the table without a match.

Example:

```
SELECT * FROM Student
FULL JOIN Course ON Student.ID = Course.StudentID;
```

Example:

```
SELECT * FROM Student
CROSS JOIN Course;
```

6. SELF JOIN: A **SELF JOIN** is a join where a table is joined with itself. It is used when we need to compare rows within the same table. To differentiate the two instances of the same table, aliases are used.

Example:

```
SELECT E1.Employee_ID, E1.Employee_Name, E2.Employee_Name AS
Manager_Name
FROM Employee E1
LEFT JOIN Employee E2 ON E1.Manager_ID = E2.Employee_ID;
```

24. What is a subquery in SQL? Provide an example.

A **subquery** in SQL is a query embedded within another query. It is used to retrieve data that will be used in the outer query. Subqueries can be used in SELECT, INSERT, UPDATE, or DELETE statements.

There are two types of subqueries:

- **Single-row subquery**: Returns a single value.
- **Multiple-row subquery**: Returns multiple values.

Example of a subquery: To find the names of students who have a higher age than the average age:

```
SELECT Name FROM Student
WHERE Age > (SELECT AVG(Age) FROM Student);
```

25. What are aggregate functions in SQL? Name a few examples.

used in conjunction with the **GROUP BY** clause. Here are a few commonly used **aggregate functions** in SQL:

1. COUNT(): Returns the number of rows or non-NULL values in a column

Example:

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

2. **SUM()**: Returns the sum of values in a numeric column.

Example:

```
SELECT SUM(Amount) FROM Orders;
```

3. AVG(): Returns the average value of a numeric column.

Example:

```
SELECT AVG(Salary) FROM Employees;
```

4. **MAX():** Returns the maximum value in a column

Example:

```
SELECT MAX(Age) FROM Student;
```

5. MIN(): Returns the minimum value in a column.

Example:

```
SELECT MIN(Salary) FROM Employees;
```

Intermediate DBMS Interview Questions

DBMS Intermediate Interview Questions dive deeper into more complex DBMS concepts and practical applications. These questions assess your ability to handle real-world database problems involving transactions, concurrency control, and schema design. Topics such as

26. What is a transaction in DBMS? What are the properties of a transaction?

A **transaction** in DBMS is a sequence of one or more SQL operations executed as a single unit of work. A transaction ensures data integrity, consistency, and isolation, and it guarantees that the database reaches a valid state, regardless of errors or system failures.

Properties of a transaction (ACID Properties):

- **Atomicity:** All operations within the transaction are completed successfully, or none are applied (i.e., the transaction is atomic).
- **Consistency:** The transaction brings the database from one valid state to another valid state.
- **Isolation:** The operations of one transaction are isolated from others; intermediate results are not visible to other transactions.
- **Durability:** Once a transaction is committed, its effects are permanent, even in the event of a system crash.

27. What is the ACID property in DBMS?

ACID stands for **Atomicity, Consistency, Isolation,** and **Durability**, which are the key properties that guarantee reliable transaction processing:

- **Atomicity:** All operations in a transaction are treated as a single unit. If one operation fails, the entire transaction fails and the database state remains unchanged.
- **Consistency:** Ensures the database starts and ends in a consistent state, with all rules and constraints enforced.
- **Isolation:** Transactions are executed independently, and the intermediate states of a transaction are invisible to other transactions.
- **Durability:** Once a transaction is committed, its changes are

A **stored procedure** is a precompiled collection of one or more SQL statements stored in the database. Stored procedures allow users to execute a series of operations as a single unit, improving performance and reusability. They can accept input parameters, perform operations, and return results.

Example:

```
CREATE PROCEDURE GetStudentDetails(IN student_id INT)
BEGIN
   SELECT * FROM Student WHERE ID = student_id;
END;
```

29. What are triggers in DBMS? Provide an example.

A **trigger** is a special kind of stored procedure that automatically executes (or "fires") in response to certain events on a table, such as insertions, updates, or deletions. Triggers are used to enforce business rules, maintain consistency, or log changes.

Example:

```
CREATE TRIGGER after_student_insert
AFTER INSERT ON Student
FOR EACH ROW
BEGIN
   INSERT INTO AuditLog (Action, StudentID, ActionTime)
   VALUES ('INSERT', NEW.ID, NOW());
END;
```

30. What is the difference between UNION and UNION ALL in SQL?

- **UNION**: Combines the result of two queries and removes duplicate rows.

Example:

```
SELECT Name FROM Students
UNION
SELECT Name FROM Teachers;  -- Removes duplicates

SELECT Name FROM Students
UNION ALL
SELECT Name FROM Teachers;  -- Does not remove duplicates
```

31. Explain the concept of indexing in DBMS.

Indexing is a technique used to speed up the retrieval of records from a table by creating a data structure that allows for faster searching. An index provides a quick lookup of data based on the values of one or more columns.

- **Types of Indexes:**
  - **Single-column index**: Created on one column.
  - **Composite index**: Created on multiple columns.
  - **Unique index**: Ensures that no two rows have the same values in the indexed columns.

32. What is a normalization form? Explain different normalization forms.

Normalization is the process of organizing a database to minimize redundancy and dependency by splitting large tables into smaller, related ones. There are several **normal forms (NF)**:

- **1NF (First Normal Form):** Ensures that each column contains atomic (indivisible) values, and each record is unique.
- **2NF (Second Normal Form):** Ensures that the table is in 1NF, and all non-key attributes are fully functionally dependent on the

- **BCNF (Boyce-Codd Normal Form):** A stricter version of 3NF, which ensures that every determinant is a candidate key.

33. What is the difference between INNER JOIN and OUTER JOIN?

Aspect	INNER JOIN	OUTER JOIN
Result Set	Returns only matching rows from both tables.	Returns all rows from one or both tables, with NULL where no match is found.
Types	Single type.	Three types: LEFT, RIGHT, FULL.
Use Case	When you need only the intersecting data.	When you need to preserve all data, even with mismatches.
Performance	Generally faster as it deals with fewer rows.	Can be slower due to handling more rows and NULL values.
Reliability	Reliable for matching data across tables.	Reliable when you need to retain all data, but can introduce NULL-related issues.

34. What is data redundancy in a database? How can it be reduced?

Data redundancy refers to the unnecessary repetition of data in a database. It can lead to inconsistencies, increased storage requirements, and maintenance challenges.

Reduction methods:

- **Eliminating duplicate data**: Using constraints like UNIQUE and PRIMARY KEY to enforce data consistency.

35. What is a deadlock in DBMS? How can it be prevented?

A **deadlock** occurs when two or more transactions are blocked because each transaction is waiting for the other to release resources. This results in a situation where none of the transactions can proceed.

Prevention techniques:

- **Lock ordering:** Ensuring that all transactions acquire locks in the same predefined order.
- **Timeouts:** Automatically rolling back transactions that have been waiting too long for resources.
- **Deadlock detection:** Periodically checking for deadlocks and aborting one of the transactions to break the cycle.

36. What is a database cursor? How is it used?

A **cursor** in DBMS is a pointer to a result set of a query. It allows for row-by-row processing of query results, which is useful when dealing with large datasets.

Types of cursors:

- **Implicit cursors:** Automatically created by the DBMS for SELECT, INSERT, UPDATE, DELETE operations.
- **Explicit cursors:** Manually created by the programmer to process query results.

Example:

```
DECLARE cursor_example CURSOR FOR
SELECT * FROM Employee;
```

There are several types of locks in DBMS to ensure data consistency when multiple transactions are involved:

- **Shared Lock (S Lock):** Allows multiple transactions to read a resource but prevents modification.
- **Exclusive Lock (X Lock):** Prevents any other transaction from reading or modifying the locked resource.
- **Intent Lock:** Signals that a transaction intends to lock a resource.
- **Update Lock (U Lock):** Used when a transaction intends to update a resource.

38. What is the difference between a clustered and non-clustered index?

- **Clustered Index:** Organizes the data in the table according to the index. There can only be one clustered index per table because the data rows can only be sorted one way.
- **Non-clustered Index:** Creates a separate structure from the table that holds pointers to the actual data rows. Multiple non-clustered indexes can be created on a table.

39. What is the importance of the COMMIT and ROLLBACK operations?

- **COMMIT:** Saves all changes made during the current transaction to the database permanently.
- **ROLLBACK:** Reverses all changes made during the current transaction, restoring the database to its previous state.
- Both operations ensure the **ACID** properties of transactions: **Atomicity** and **Durability**.

40. What is the difference between a superkey and a candidate key?

- **Candidate Key:** A minimal superkey that uniquely identifies a row, with no redundant attributes. A table can have multiple candidate keys, and one is chosen as the **Primary Key**.

41. What are the different types of constraints in DBMS? Give examples.

Constraints in DBMS are rules applied to the data in a database to ensure its accuracy and integrity. The most common types of constraints are:

- **NOT NULL**: Ensures that a column cannot have NULL values. **Example**: `Name VARCHAR(50) NOT NULL;`
- **PRIMARY KEY**: Uniquely identifies each record in a table. It ensures that no duplicate rows exist and that no NULL values are allowed. **Example**: `ID INT PRIMARY KEY;`
- **FOREIGN KEY**: Ensures referential integrity between two tables by linking a column in one table to the primary key in another table. **Example**: `BranchCode INT FOREIGN KEY REFERENCES Branch(BranchCode);`
- **UNIQUE**: Ensures that all values in a column are distinct. Unlike the primary key, it allows NULL values. **Example**: `Email VARCHAR(100) UNIQUE;`
- **CHECK**: Ensures that values in a column satisfy a specific condition. **Example**: `Age INT CHECK (Age >= 18);`
- **DEFAULT**: Assigns a default value to a column if no value is provided during insertion. **Example**: `Status VARCHAR(10) DEFAULT 'Active';`

42. Explain the difference between a primary key and a unique key.

Primary Key:

- Uniquely identifies each record in a table.

Unique Key:

- Ensures that all values in a column (or a set of columns) are unique across all rows.
- Can contain `NULL` values (unlike a primary key).
- A table can have **multiple** unique keys.

43. What is referential integrity in DBMS?

Referential Integrity ensures that relationships between tables are maintained correctly. It requires that the foreign key in one table must match a primary key or a unique key in another table (or be NULL). This ensures that data consistency is maintained, and there are no orphan records in the database.

Example: In the `Orders` table, if the `CustomerID` is a foreign key, it should match a valid `CustomerID` in the `Customers` table or be NULL.

44. How does DBMS handle concurrency control?

Concurrency control ensures that database transactions are executed in a way that prevents conflicts, such as data inconsistency, when multiple transactions are executed simultaneously. DBMS uses the following techniques:

- **Locking**: Transactions acquire locks (shared or exclusive) on the data to prevent other transactions from modifying it while one transaction is in progress. **Types of Locks**: Shared locks (S-lock) and exclusive locks (X-lock).
- **Timestamp Ordering**: Assigns a unique timestamp to each transaction and uses these timestamps to determine the order in which transactions should be executed.
- **Optimistic Concurrency Control**: Transactions are executed without locking data, but before committing, the system checks whether there were conflicts with other transactions.

ensure serializability.

45. What are the advantages of using foreign keys in DBMS?

- **Enforcing Data Integrity**: Ensures that the value of a foreign key matches a valid primary key or unique key, maintaining consistency.
- **Referential Integrity**: Prevents orphaned records in the database by enforcing valid relationships between tables.
- **Easy Data Maintenance**: Helps with cascading updates and deletions, meaning changes in the referenced table can automatically propagate to the referencing table (if configured with `ON UPDATE CASCADE` or `ON DELETE CASCADE`).
- **Improved Query Efficiency**: With foreign keys, database queries that join related tables are more efficient and meaningful.

46. What is a transaction log in DBMS?

A **transaction log** is a record that keeps track of all transactions executed on a database. It ensures that changes made by transactions are saved, and in case of a system failure, the log can be used to recover the database to its last consistent state. The transaction log contains:

- The details of each transaction (e.g., start, commit, rollback).
- Information about data modifications (insertions, updates, deletions).
- Details about the data before and after the change.

47. What is a materialized view in DBMS?

A **materialized view** is a database object that contains the results of a query. Unlike a regular view, which is a virtual table (it doesn't store data), a materialized view stores data physically, improving query performance by precomputing and storing results.

same data is frequently queried.

Example:

```
CREATE MATERIALIZED VIEW SalesSummary AS
SELECT Product, SUM(Quantity) FROM Sales GROUP BY Product;
```

48. What are the differences between an ER diagram and a relational schema?

Entity-Relationship Diagram (ERD):

- A **conceptual blueprint** that models entities, relationships, and attributes of the database. It visually represents the structure of the database.
- Used in the database design phase to understand how data entities relate to each other.

Relational Schema:

- A **logical schema** that defines the structure of a relational database, including tables, columns, relationships, and constraints.
- Represents how data is physically organized in tables with constraints such as primary keys, foreign keys, and data types.

49. What is the purpose of the GROUP BY clause in SQL?

The **GROUP BY** clause is used in SQL to group rows that have the same values in specified columns into summary rows, often with aggregate functions like `COUNT`, `SUM`, `AVG`, `MIN`, or `MAX`. It is typically used to organize data for reporting or analysis.

Example: This groups the employees by department and counts the number of employees in each department.

```
SELECT Department, COUNT(*) FROM Employees
```

A **stored function** is a set of SQL statements that can be executed in the database. It accepts input parameters, performs some logic, and returns a value. Stored functions are similar to stored procedures but differ in that they must return a value.

Example:

```
CREATE FUNCTION GetEmployeeSalary(EmployeeID INT)
RETURNS DECIMAL(10,2)
BEGIN
   DECLARE salary DECIMAL(10,2);
   SELECT Salary INTO salary FROM Employee WHERE ID =
EmployeeID;
   RETURN salary;
END;
```

Advanced DBMS Interview Questions

DBMS Advanced Interview Questions focus on in-depth knowledge and problem-solving skills for complex database scenarios. This category evaluates your ability to handle large-scale data, optimize query performance, and understand advanced DBMS topics like distributed databases, indexing strategies, and concurrency control mechanisms. Being proficient in these areas is crucial for roles involving high-performance databases and large-scale systems.

51. What are the various types of normalization techniques? Explain with examples.

Normalization is the process of organizing data in a database to eliminate redundancy and improve data integrity. The primary goal is to minimize the chances of anomalies when performing operations like insertions, deletions, and updates. There are several levels or **normal forms** in normalization:

1. **First Normal Form (1NF)**: Ensures that the data in the table is

Example:

```
Student (ID, Name, Subjects)
```

2. Second Normal Form (2NF): 2NF is achieved when the table is in 1NF and all non-key attributes are fully dependent on the primary key.

Example: Consider the following table

```
StudentCourse (StudentID, CourseID, InstructorName)
```

Here, `InstructorName` depends on `CourseID`, not on `StudentID`. To make it 2NF, we separate the data:

```
Student (StudentID, Name)
Course (CourseID, InstructorName)
Enrollment (StudentID, CourseID)
```

3. Third Normal Form (3NF): A table is in 3NF if it is in 2NF and there is no transitive dependency (non-key attributes depend on other non-key attributes).

Example:

```
Employee (EmpID, EmpName, DeptID, DeptName)
```

Here, `DeptName` depends on `DeptID`. To bring this into 3NF:

```
Employee (EmpID, EmpName, DeptID)
Department (DeptID, DeptName)
```

4. Boyce-Codd Normal Form (BCNF): A stricter version of 3NF where every determinant is a candidate key.

Example:

```
Student (StudentID, CourseID, InstructorID)
```

```
Instructor (InstructorID, CourseID)
Enrollment (StudentID, CourseID)
```

52. What are the different phases of the DBMS query processing cycle?

The DBMS query processing cycle consists of several phases that transform the high-level query (SQL) into executable actions:

1. **Parsing**: The SQL query is parsed to check its syntax and semantics. The DBMS ensures that the query is valid according to the SQL syntax and the database schema.
2. **Translation**: The query is translated into an internal form, such as a relational algebra expression or an execution plan.
3. **Optimization**: The DBMS optimizes the query to determine the most efficient execution plan, considering factors like indexes, joins, and available resources.
4. **Execution**: The optimized query plan is executed by the query processor, which accesses the data from the database and returns the results.

53. What are the different types of backups in DBMS?

There are several types of backups in DBMS:

- **Full Backup**: A full backup copies the entire database, including all data and the database structure. It is the most comprehensive but can take up a lot of storage and time.
- **Incremental Backup**: An incremental backup only copies the data that has changed since the last backup (either full or incremental). This saves space and time but requires the restoration of the full backup and all incremental backups.
- **Differential Backup**: A differential backup copies all changes made

- **Transaction Log Backup**: A transaction log backup copies the transaction log, which records all transactions performed on the database. This allows for point-in-time recovery.

54. What is the use of the "WITH CHECK OPTION" in SQL views?

The **"WITH CHECK OPTION"** is used when creating a view in SQL to ensure that any insert or update operation on the view must adhere to the conditions defined in the view's WHERE clause. If the inserted or updated data violates these conditions, the operation will be rejected.

Example: Here, if a user tries to insert or update a `Student` record with a status other than 'Active', the operation will fail.

```
CREATE VIEW ActiveStudents AS
SELECT * FROM Students WHERE Status = 'Active'
WITH CHECK OPTION;
```

55. Explain the concept of a B-tree and B+ tree in DBMS.

B-tree (Balanced Tree):

- A **B-tree** is a self-balancing tree data structure that maintains sorted data and allows searches, insertions, deletions in logarithmic time.
- B-trees are used in databases and file systems to store large amounts of data. All nodes in a B-tree can have multiple children, which increases the efficiency of searching.
- Stores data in both internal and leaf nodes.

B+ tree:

- A **B+ tree** is an extension of the B-tree and is widely used in databases for indexing. It differs in that it has a linked list at the leaf

or pointers to the data.
- Stores data only in the leaf nodes and uses the internal nodes for indexing.

56. What is a hashing technique in DBMS? How does it work?

A **hashing technique** in DBMS is used to map data (such as a key) to a fixed-size value or address, using a hash function. It is primarily used for quick data retrieval, particularly in hash indexes or hash tables. **Example**: A hash function might map a `StudentID` of `123` to a bucket index of `3`. The student's record would be stored in the corresponding bucket.

How it works:

1. A hash function takes the key (e.g., a record's ID) and calculates a **hash value**.
2. This hash value determines the **bucket** or **slot** where the data is stored.
3. When searching for a record, the hash function is applied again to the key to find the corresponding bucket.

57. What is the difference between a trigger and a stored procedure?

Trigger:

- A **trigger** is an automatic action executed by the DBMS when a specific event occurs on a table, such as an `INSERT`, `UPDATE`, or `DELETE`.
- It cannot be invoked manually and is tied to a specific event.

Stored Procedure:

- A **stored procedure** is a precompiled set of SQL statements that can be executed explicitly by a user or application.

58. Explain the concept of data partitioning in DBMS.

Data partitioning is the process of dividing large datasets into smaller, more manageable segments (partitions) to improve performance, scalability, and availability. Each partition can be stored and processed separately.

Types of partitioning:

1. **Horizontal Partitioning**: Divides data by rows. For example, splitting data by time range (e.g., 2020 data in one partition, 2021 in another).
2. **Vertical Partitioning**: Divides data by columns. For example, putting frequently accessed columns in one partition and less frequently accessed columns in another.
3. **Range Partitioning**: Data is divided based on a range of values (e.g., age groups, date ranges).
4. **Hash Partitioning**: Data is distributed across partitions based on a hash value derived from a key column.

59. What is the role of the DBMS in handling data integrity and security?

The **DBMS** plays a critical role in ensuring:

- **Data Integrity**: Through constraints like **Primary Keys**, **Foreign Keys**, and **Check Constraints**, the DBMS ensures data consistency and accuracy.
- **Data Security**: DBMS systems provide user authentication, access control, and encryption mechanisms to protect data from unauthorized access and breaches. It also supports role-based access control (RBAC), ensuring that only authorized users can perform certain actions on the data.

60. How is DBMS different from a file-based system?

Aspect	DBMS	File-based System
Data Organization	Data is stored in tables (relations) with structured schemas, supporting complex queries and relationships.	Data is stored in flat files without any relationship between data.
Data Redundancy	Minimizes redundancy through normalization.	Data redundancy is common as data may be duplicated across different files.
Data Integrity	Ensures data integrity through constraints (e.g., primary keys, foreign keys, and check constraints).	Data integrity is hard to enforce, as files don't have built-in integrity checks.
Security	Provides advanced security features like user authentication, access control, and encryption.	Security is managed at the file system level, which is less robust than DBMS security features.
Concurrency Control	Handles concurrent access using locking mechanisms, ensuring data consistency.	No built-in concurrency control; data corruption may occur when multiple users access the same file.
Data Access	Supports complex querying and transaction management (e.g., SQL).	Data access is limited to basic file operations like reading, writing, and appending.

Aspect	DBMS	File-based System
Backup and Recovery	Automatic backup and recovery mechanisms, often integrated into the system.	Backup and recovery mechanisms are manual and may not be as robust.
Scalability	Easily scalable to handle large datasets and multiple users.	Not as scalable; managing large amounts of data and multiple users is difficult.
Maintenance	Centralized maintenance, with tools for optimization, tuning, and troubleshooting.	Maintenance is usually handled at the file system level, which may require manual intervention.
Transaction Management	Supports ACID (Atomicity, Consistency, Isolation, Durability) properties for reliable transaction management.	No built-in support for transactions or ACID properties, making it prone to errors during data modification.

Conclusion

Preparing for a **DBMS interview** requires a strong understanding of both fundamental and advanced concepts. From basic database operations to handling large-scale data with advanced indexing strategies, mastering these DBMS interview questions will prepare you for a variety of challenges you may encounter in your career. By practicing these questions, you'll enhance your problem-solving skills, boost your confidence, and be ready to tackle any database-related

1. What is DevOps and how does it relate to Cloud Engineering?

DevOps is a culture that emphasizes collaboration between development and operations teams. In Cloud Engineering, it allows for faster deployment, continuous integration, and more reliable systems by utilizing cloud resources to automate processes and enhance scalability.

Example:

DevOps promotes synergy between development and operations, enabling continuous integration and deployment. In the cloud context, it leverages automation tools to streamline workflows and improve system reliability.

2. Can you explain Infrastructure as Code (IaC) and its benefits?

Infrastructure as Code (IaC) is the practice of managing and provisioning computing infrastructure through machine-readable scripts rather than physical hardware configuration. The benefits include consistency, repeatability, faster deployments, and reduced human error, enabling teams to manage infrastructure efficiently.

Example:

IaC allows infrastructure management through code, enhancing consistency and repeatability. It speeds up deployment processes and minimizes human error, crucial for scaling applications in cloud environments.

3. What tools do you use for Continuous Integration/Continuous Deployment (CI/CD)?

I utilize tools like Jenkins, GitLab CI, and CircleCI for CI/CD processes. These tools help automate the testing and deployment stages, ensuring faster delivery of features while maintaining high-quality standards through automated testing.

Example:

I use Jenkins and GitLab CI for CI/CD, automating testing and deployment. This approach enhances delivery speed while ensuring quality through consistent automated testing procedures.

4. How do you monitor cloud infrastructure?

I monitor cloud infrastructure using tools like CloudWatch, Prometheus, and Grafana. These tools provide real-time insights into system health, performance metrics, and alerting mechanisms, allowing for proactive management and quick resolution of issues.

Example:

I leverage CloudWatch and Grafana for monitoring cloud infrastructure. They provide real-time metrics and alerts, enabling proactive management and rapid issue resolution to maintain system performance.

5. Describe a challenging project you managed in a cloud environment.

I led a project migrating legacy applications to AWS. The challenge was minimizing downtime while ensuring data integrity. I implemented a phased migration strategy with comprehensive testing, allowing for a successful transition with minimal disruption to users.

Example:

I managed a complex migration of legacy apps to AWS, focusing on minimizing downtime. I employed a phased approach with extensive testing, ensuring a smooth transition and data integrity throughout the process.

6. How do you handle security in cloud environments?

I prioritize security by implementing best practices such as IAM policies, encryption, and regular audits. Additionally, I utilize tools like AWS Config and Azure Security Center to monitor and manage security configurations and compliance.

Example:

I ensure cloud security through stringent IAM policies and encryption. Regular audits and tools like AWS Config help monitor compliance and secure configurations effectively.

7. What is your experience with containerization and orchestration?

I have extensive experience with Docker for containerization and Kubernetes for orchestration. These technologies streamline application deployment, scalability, and management, allowing for efficient resource utilization and improved application consistency across environments.

Example:

I use Docker for containerization and Kubernetes for orchestration, enabling efficient application deployment and management, ensuring scalability, and enhancing resource utilization across various environments.

8. Explain how you ensure high availability in cloud applications.

To ensure high availability, I implement load balancers, auto-scaling groups, and multi-region deployments. These strategies minimize downtime and distribute traffic effectively, ensuring that applications remain accessible even during peak loads or failures.

Example:

I ensure high availability by utilizing load balancers, auto-scaling groups, and multi-region deployments, which effectively distribute traffic and minimize downtime, maintaining application accessibility.

9. How do you ensure high availability in a cloud environment?

To ensure high availability, I utilize multi-region deployments and auto-scaling groups. I also implement load balancers to distribute traffic evenly and regularly test disaster recovery plans to minimize downtime.

Example:

I set up a multi-region architecture with load balancers to manage traffic and deployed auto-scaling groups to handle peak loads, ensuring that our applications remain accessible even during failures.

10. What tools do you use for CI/CD in a cloud environment?

I commonly use Jenkins for CI/CD pipelines along with Git for version control. Integration with cloud providers like AWS CodePipeline helps automate deployments and manage infrastructure as code.

Example:

I integrated Jenkins with Git and AWS CodePipeline, enabling automated testing and deployment, which significantly reduced our release cycle time while ensuring high quality.

11. How do you manage secrets in a cloud environment?

I use tools like AWS Secrets Manager or HashiCorp Vault to securely store and manage secrets. This practice helps prevent hardcoding sensitive information in codebases while allowing controlled access.

Example:

I implemented AWS Secrets Manager to store API keys and database credentials, ensuring that developers had secure access without exposing sensitive data in our repositories.

12. Describe your experience with containerization technologies.

I have extensive experience with Docker for creating containerized applications and Kubernetes for orchestration. This experience allows me to efficiently manage microservices and maintain consistency across environments.

Example:

I used Docker to containerize applications, streamlining deployment across environments, and leveraged Kubernetes to manage container orchestration, which simplified scaling and updates.

13. How do you monitor applications in a cloud environment?

I use monitoring tools like Prometheus and Grafana for real-time metrics, combined with logging solutions like ELK Stack. This approach allows for proactive monitoring and quick troubleshooting.

Example:

I set up Prometheus to collect metrics and Grafana for visualization, enabling us to quickly identify performance issues and optimize our services based on real-time data.

14. Explain your approach to handling system failures.

I follow a systematic approach, starting with root cause analysis to identify failures, followed by implementing fixes and updating documentation. I also review our incident response plans to improve future resilience.

Example:

After a system failure, I conducted a root cause analysis, documented findings, and updated our incident response plan, which helped prevent similar issues from recurring.

15. What is Infrastructure as Code (IaC) and why is it important?

Infrastructure as Code allows the management of infrastructure through code, enabling automation, consistency, and version control. This practice reduces manual errors and accelerates deployments in cloud environments.

Example:

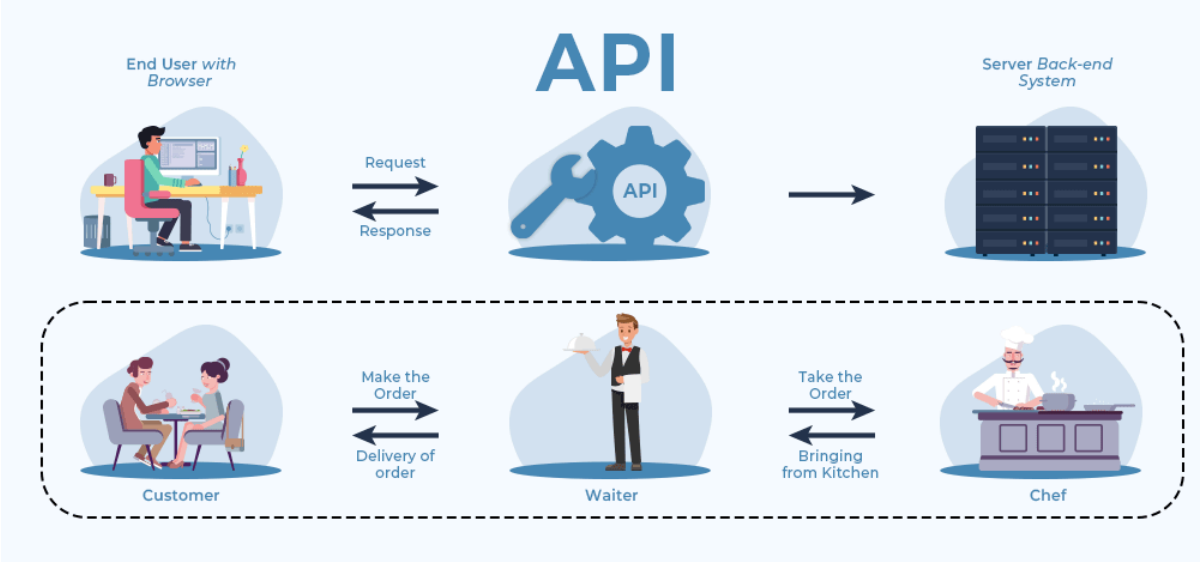
By using Terraform for IaC, I automated our infrastructure setup, which reduced deployment times and ensured that our environments were consistently configured.

What is an API (Application Programming Interface)

Last Updated : 21 Jul, 2025

An API is a set of rules that allow different software applications to communicate with each other .Think of it as a **bridge** that connects two systems—such as a client and a server—and enables them to work together seamlessly.

To understand it better, imagine you're at a restaurant: the **waiter** (API) takes your order (request), gives it to the **chef** (server), and then brings the prepared food (response) back to your table. Similarly, when you search for a course on a website, your request goes through an API, which then fetches the data from the database and sends it back as a response.



Today, over **80% of modern web applications** rely on APIs to fetch data, integrate with third-party services, or enable features like login, payments, or real-time updates. APIs play a crucial role in building

Why Do We Need APIs?

APIs help developers to create software programs more easily. Instead of writing complex code from scratch, they can call APIs that already provide the functions they need. For example, if a developer wants to display a weather report, they can use an API to get the data instead of creating the entire system to gather weather data themselves.

APIs are also crucial in building modern websites, where heavy data transfers happen between the client (user) and the server.

How Do APIs Work?

APIs work in a simple step-by-step process:

- **Request:** A client (user) sends a request through the API's URI (Uniform Resource Identifier).
- **Processing:** The API forwards the request to the server.
- **Response:** The server processes the request and sends the response back to the API.
- **Delivery:** The API returns the server's response to the client.

Think of this as a client-server architecture: the client sends a request, the server processes it, and the API acts as the messenger.curity threats. To provide additional security layers to the data, HTTP headers, query string parameters, or cookies are used.

Types of API Architectures:

1. **REST (Representational State Transfer) :**A simple, flexible API architecture that uses HTTP methods (GET, POST, PUT, DELETE) for communication.
2. **SOAP (Simple Object Access Protocol) :**A more rigid protocol that requires XML-based messaging for communication.

Both define a standard communication protocol for the exchange of

An API acts as an interface that allows proper communication between two programs whereas a web application is a network-based resource responsible for completing a single task. Also, it's important to know that **"All web services are APIs, but not all APIs are web".**

The difference between an API and a web application is that API allows two-way communication and web applications are just a way for users to interact through a web browser. A web application may have an API to complete the requests.

Types of APIs

There are three basic forms of API -

1. WEB APIs

A **Web API** also called Web Services is an extensively used API over the web and can be easily accessed using the HTTP protocols. A Web **application programming interface is** an open-source interface and can be used by a large number of clients through their phones, tablets, or PCs.

2. LOCAL APIs

In this type of API, the programmers get the local middleware services. TAPI (Telephony Application Programming Interface), and .NET are common examples of Local APIs.

3. PROGRAM APIs

It makes a remote program appear to be local by making use of RPCs (Remote Procedural Calls). SOAP is a well-known example of this type of API.

Few other types of APIs:

- **REST (Representational State Transfer):** It makes use of HTTP to GET, POST, PUT, or DELETE data. It is basically used to take advantage of the existing data.
- **JSON-RPC:** It uses JSON for data transfer and is a lightweight remote procedural call defining a few data structure types.
- **XML-RPC:** It is based on XML and uses HTTP for data transfer. This API is widely used to exchange information between two or more networks.

What are REST APIs?

REST stands for Representational State Transfer, and follows the constraints of REST architecture allowing interaction with RESTful web services. It defines a set of functions (GET, PUT, POST, DELETE) that clients use to access server data. The functions used are:

- GET (retrieve a record)
- PUT (update a record)
- POST (create a record)
- DELETE (delete the record)

Its main feature is that REST API is stateless, i.e., the servers do not save clients' data between requests.

What is a Web API?

Web API Is simply an API for the web. It is an API that can be accessed using the HTTP protocol. It can be built using Java, .nET, etc. It is implemented to extend the functionality of a browser, simplify complex functions, and provide easy syntax to complex code.

The four main types of web APIs are:

- Open API
- Partner API
- Internal API
- Composite API

SOAP vs. REST

SOAP	REST
SOAP (Simple Object Access Protocol) is a **protocol** with specific requirements like XML messaging	REST (Representational State Transfer) is a set of guidelines **(architectural style)** offering flexible implementation
Heavier and needs more bandwidth	Lightweight and needs less bandwidth
It defines its own security	It inherits security from the underlying transport
It permits XML-based data format only	It permits different data formats such as plain text, HTML, XML, JSON, etc.
SOAP calls cannot be cached	REST calls can be cached

Also, the major difference is that SOAP cannot make use of REST whereas REST can make use of SOAP. You can also read about the **difference between REST API and SOAP API**

What is API (Application Programming Interface) Integration?

API (Application Programming Interface) Integration is the connection between two or more applications, via APIs, letting you exchange data. It is a medium through which you can share data and communicate with each other by involving APIs to allow web tools to communicate. Due to the rise in cloud-based products, API integration has become very important.

API (Application Programming Interface) testing is a kind of software testing that analyzes an API in terms of its functionality, security, performance, and reliability. It is very important to test an API so as to check whether it's working as expected or not. If not, again changes are made in the architecture and re-verified.

APIs are the center of software development to exchange data across applications. The API testing includes sending requests to single/multiple API endpoints and validating the response. It focuses majorly on business logic, data responses and security, and performance bottlenecks.

Types of Testing:

- Unit Testing
- Integration Testing
- Security Testing
- Performance Testing
- Functional Testing

Must Read: API Testing in Software Testing

API Testing Tools:

- Postman
- Apigee
- JMeter
- Ping API
- Soap UI
- vREST

How to Create APIs?

Creating an API is an easy task unless you are very well clear on the basic concepts. It's an iterative process (based on feedback) that just

- Develop (Implement the code) and Test API
- Monitor its working and work on feedback

Must Read: Tips for Building an API

Restrictions of Using APIs

When an API (Application Programming Interface) is made it's not really released as software for download and it has some policies governing its use or restricting its use to everyone, usually, there are three main types of policies governing APIs, are:

- **Private:** These APIs are only made for a single person or entity (like a company that has spent the resources to make it or bought it).
- **Partner:** Just like the name it gives the authority to use APIs to some partners of entities that own APIs for their private use.
- **Public:** You should be aware of them cause you can only find these APIs in the market for your own use if you don't own specific API access from some entity that owns private these APIs for their private use. An example of a Public API is 'Windows API' by Microsoft for more public APIs you can visit this GitHub repository -> https://github.com/public-apis/public-apis .

Advantages of APIs

- **Efficiency:** API produces efficient, quicker, and more reliable results than the outputs produced by human beings in an organization.
- **Flexible delivery of services:** API provides fast and flexible delivery of services according to developers' requirements.
- **Integration:** The best feature of API is that it allows the movement of data between various sites and thus enhances the integrated user experience.
- **Automation:** As API makes use of robotic computers rather than humans, it produces better and more automated results.
- **New functionality** : While using API the developers find new tools

- **Cost:** Developing and implementing API is costly at times and requires high maintenance and support from developers.
- **Security issues:** Using API adds another layer of surface which is then prone to attacks, and hence the security risk problem is common in APIs.

Conclusion

By now, you must have had a clear idea of **What is API?** it's working, types, testing tools used, etc. After understanding these concepts, you can try working on them by implementing some of the concepts in projects. Not just theoretical knowledge, you must also have a practical idea of it by working on it. Developers must have a deep understanding of APIs in order to implement them.

Top characteristics of software are:

- **Functionality:** It refers to the software performance compared to the purpose it was created for.
- **Reliability:** It is a characteristics of software that refers to its ability to perform what it was designed to do accurately and consistently over time.
- **Usability (User-friendly):** It refers to the extent to which the software can be used with ease. The amount of effort or time required to learn how to use the software.
- **Efficiency:** It refers to the ability of the software to use system resources in the most effective and efficient manner.
- **Flexibility:** It refers to how simple it is to improve and modify the software.
- **Maintainability:** It refers to how easily a software system can be modified to add feature, improve speed, or repair faults.
- **Portability:** It refers to how well the software can work on different platforms or situations without making major modifications.
- **Integrity:** It refers to how well the software maintains the accuracy and consistency of data throughout its cycle.

For more details please refer to the following article Characteristics of Software.

2. What are the Various Categories of Software?

The software is used extensively in several domains including hospitals, banks, schools, defense, finance, stock markets, and so on. It can be categorized into different types:

1. Based on Application

1. System Software- This type of software helps manage the hardware of your computer, like an operating system that controls how the computer works.
2. Application Software- These are the programs we use every day, such as word processors or music players, to perform specific tasks.
3. Web Applications Software- These are programs you access via a web browser, like checking your email or managing your bank account online.
4. **Embedded Software-**This software is built into devices like cars, home

5. **Reservation Software**: Used to manage bookings and reservations, such as for hotels, flights, or restaurant tables.
6. **Business Software-** Designed to help businesses run smoothly, this software includes things like CRM (Customer Relationship Management) tools and ERP (Enterprise Resource Planning) systems.
7. **Artificial Intelligence Software-** These programs use machine learning to mimic human intelligence and perform tasks like recommendations or chatbots.
8. **Scientific Software-** Software used by researchers for tasks like simulations, analyzing data, or modeling experiments.

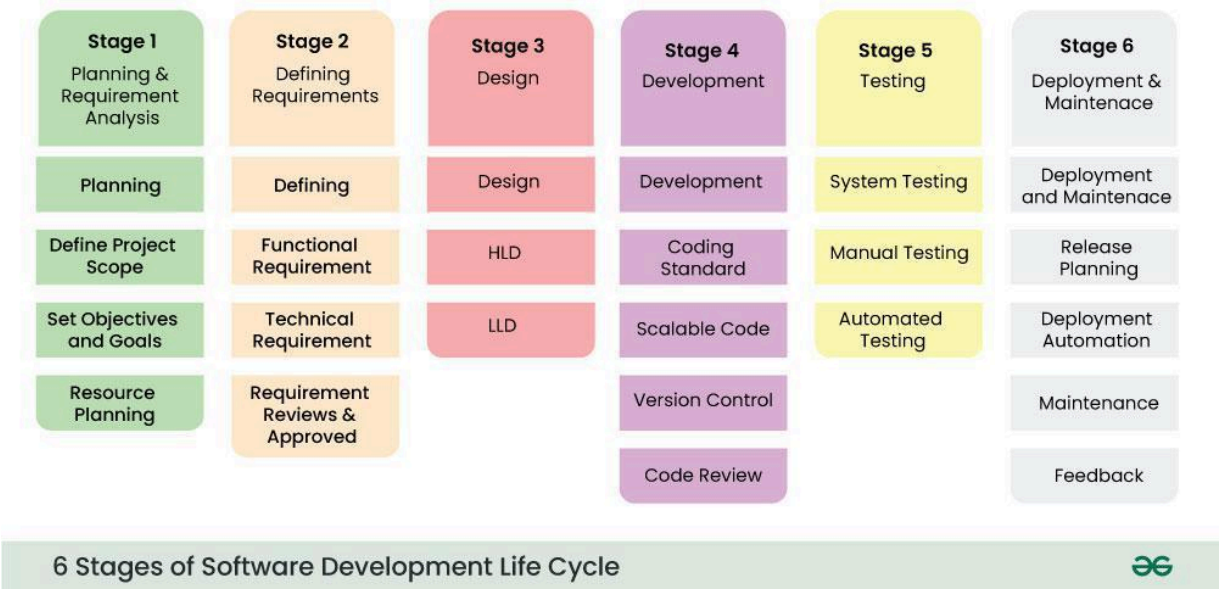
2. **Based on Copyright**

1. **Commercial Software**: This is software you pay for, usually with a license that limits how it can be used, shared, or changed.
2. **Shareware Software:** This software is available for free for a limited time or with limited features, with the hope that users will eventually pay for the full version.
3. **Freeware Software**: Software that's completely free to use, but the source code is usually not available for you to change or share.
4. **Public Domain Software**: This software is free for everyone to use, modify, and share without any copyright restrictions.

For more details please refer to the following article Classifications of Software.

3. Explain SDLC and its Phases?

SDLC stands for **Software Development Life Cycle**. It is a process followed for software building within a software organization**.** SDLC consists of a precise plan that describes how to develop, maintain, replace, and enhance specific software. The life cycle defines a method for improving the quality of software and the all-around development process.



6 Stages of Software Development Life Cycle

Phases of SDLC: Following are the phases of SDLC:

1. **Planning and Requirement Analysis**: This is where we figure out the project's goals, understand what users need, and set clear expectations for what the software should do.
2. **Defining Requirements**: Here, we get into the specifics—laying out exactly what the software needs to do (functional) and how well it should do it (non-functional).
3. **Designing Architecture**: At this stage, we build a blueprint for the software, deciding on the overall structure and the technical details to make sure it meets the requirements.
4. **Developing Product**: This is where the coding happens, turning the design into a working product by writing the actual software.
5. **Product Testing and Integration**: Now, we test the software for bugs, check that everything works together smoothly, and make sure all the parts are integrated properly.
6. **Deployment and Maintenance of Products**: Finally, the software is deployed for use, and we continue to monitor and maintain it to fix any issues and keep it running smoothly over time.

For more details, please refer to the following article Software Development Life Cycle.

4. What are different SDLC Models Available?

1. Waterfall Model- This is a straightforward, step-by-step process where each phase must be finished before moving on to the next. It's ideal for projects with clear and fixed requirements from the start.
2. V-Model- Also called the Verification and Validation model, this approach focuses on testing each part of the development process as you go. For every phase of development, there's a corresponding testing phase.
3. Incremental Model- Instead of developing everything at once, this model breaks the software into smaller, manageable pieces or "increments." These pieces are developed and delivered one at a time, allowing users to get a working version early on.
4. RAD Model - This model focuses on getting the software up and running quickly through prototypes and constant user feedback. It works well for projects with clear user requirements and tight deadlines.
5. Iterative Model - In this approach, development happens in cycles. After each cycle, a version of the software is tested, feedback is gathered, and improvements are made in the next cycle.
6. Spiral Model - This model combines design and prototyping while focusing heavily on risk assessment. It involves repeating phases of planning, design, development, and testing with each loop, addressing any risks along the way.
7. Prototype model- Here, a working version of the product (prototype) is created early on. This allows users to interact with it and give feedback, which helps shape the final product.
8. Agile Model- Agile is all about flexibility. Development happens in short bursts, or sprints, with constant feedback from the customer. This model allows for changes at any stage, making it adaptable and responsive to evolving needs.

For more details, please refer to the following article Top Software Development Models (SDLC) Models.

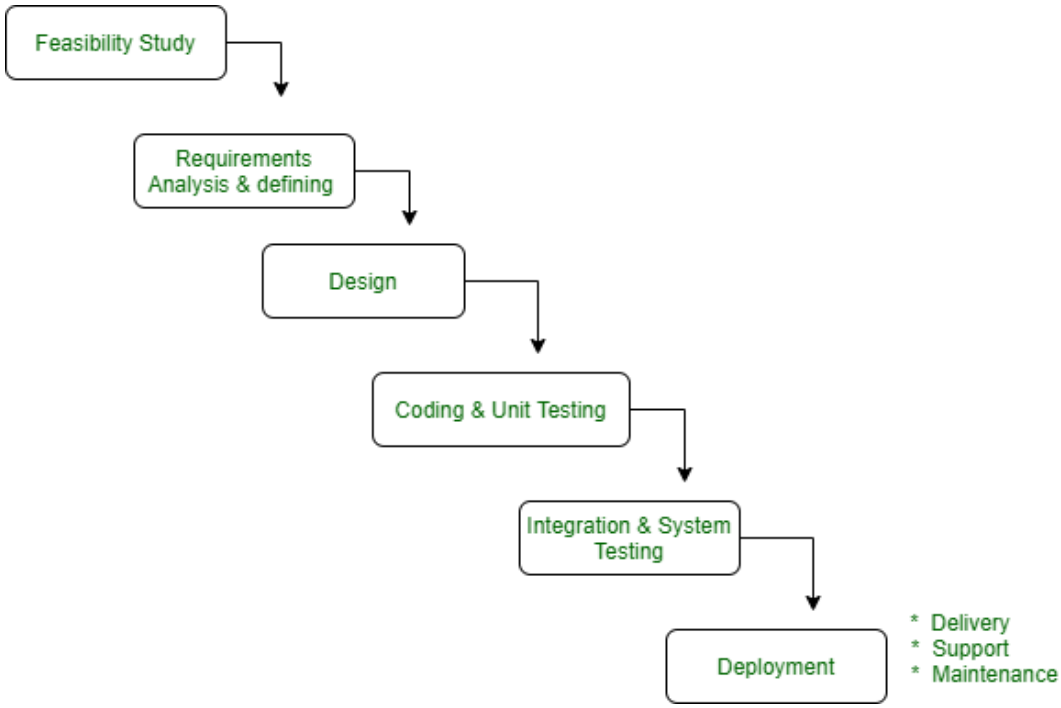
5. What is the Waterfall Method and What are its Use Cases?

The waterfall model is a software development model used in the context of large, complex projects, typically in the field of information technology. It is characterized by a structured, sequential approach to project management and software development.

1. **Requirements Gathering and Analysis**: This is the first step where you gather all the details about what the client needs and analyze them to understand exactly what the software should do.
2. **Design Phase**: Once you have a clear idea of the requirements, you move on to designing how the system will work. This is where you figure out the structure and how different parts of the software will fit together.
3. **Implementation and Unit Testing**: Now it's time to start coding. In this phase, developers write the actual software and test each small piece (or module) to make sure it works properly on its own.
4. **Integration and System Testing**: After all the individual modules are built and tested, they're put together to form the complete system. The whole system is tested to ensure everything works as expected when all the parts are connected.
5. **Deployment**: Once the software is tested and ready, it's launched and made available to users in the real world.
6. **Maintenance**: After the software is up and running, it enters the maintenance phase. This involves fixing any issues users find and making updates as needed to keep everything running smoothly.

Use Case of Waterfall Model

- Requirements are clear and fixed that may not change.
- There are no ambiguous requirements (no confusion).
- It is good to use this model when the technology is well understood.
- The project is short and cast is low.
- Risk is zero or minimum.



Waterfall Model

For more details, please refer to the following article Waterfall Model.

6. What is Black Box Testing?

The black box test (also known as the conducted test/ closed box test/ opaque box test) is software testing technique. In this technique, tester does not care about the internal knowledge or implementation details but rather focuses on validating the functionality based on the provided specifications or requirements. The name "black box" refers to the idea that the internal workings are hidden from the tester's view.

For more details, please refer to the following article Software Engineering - Black Box Testing.

7. What is White Box Testing?

White Box Testing is a method of analyzing the internal structure, data structures used, internal design, code structure, and behavior of software, as well as functions such as black-box testing. Also called glass-box test or clear box test or structural test.

For more details, please refer to the following article Software Engineering - White Box Testing.

Following are the differences between Alpha and Beta Testing:

Alpha Testing	Beta Testing
Alpha testing involves both white box and black box testing.	Beta testing commonly uses black-box testing.
Alpha testing is performed by testers who are usually internal employees of the organization.	Beta testing is performed by clients who are not part of the organization.
Alpha testing is performed at the developer's site.	Beta testing is performed at the end-user, the of the product.
Reliability and security testing are not checked in alpha testing.	Reliability, security, and robustness are checked during beta testing.
Alpha testing ensures the quality of the product before forwarding it to beta testing.	Beta testing also concentrates on the quality of the product but collects the user's time-long input on the product and ensures that the product is ready for real-time users.
Alpha testing requires a testing environment or a lab.	Beta testing doesn't require a testing environment or lab.
Alpha testing may require a real-time long execution cycle.	Beta testing requires only a few weeks of execution.
Developers can immediately address the critical issues or fixes in alpha testing.	Most of the issues or feedback collected from the beta testing will be implemented in future versions of the product

For more details, please refer to the following article Alpha Testing and Beta Testing.

Debugging is the process of identifying and resolving errors, or bugs, in a software system. It is an important aspect of software engineering because bugs can cause a software system to malfunction, and can lead to poor performance or incorrect results. Debugging can be a time-consuming and complex task, but it is essential for ensuring that a software system is functioning correctly.

> *For more details, please refer to the following article What is Debugging?*

10. What is a Feasibility Study?

The Feasibility Study in Software Engineering is a study that analyze whether a proposed software project is practical or not. It early detects the potential issues, analyzes technological possibilities, and determines the project's financial and operational viability. This decreases the chance of project failure that also save time and money.

> *For more details, please refer to the following article Types of Feasibility Study in Software Project Development article.*

11. What is a Use Case Diagram?

A use case diagram is a behavior diagram and visualizes the observable interactions between actors and the system under development. The diagram consists of the system, the related use cases, and actors and relates these to each other:

- **System**: What is being described?
- **Actor**: Who is using the system?
- **Use Case**: What are the actors doing?

> *For more details, please refer to the following article use case diagram.*

12. What is the difference between Verification and Validation?

Here are the difference between Verification and Validation

Verification	Validation
Verification is a static practice of verifying documents, design, code, black-box, and programs human-based.	Validation is a dynamic mechanism of validation and testing the actual product.
It does not involve executing the code.	It always involves executing the code.
It is human-based checking of documents and files.	It is computer-based execution of the program.
Verification uses methods like inspections, reviews, walkthroughs, and Desk-checking, etc.	Validation uses methods like black box (functional) testing, gray box testing, and white box (structural) testing, etc.
Verification is to check whether the software conforms to specifications.	Validation is to check whether the software meets the customer's expectations and requirements.
It can catch errors that validation cannot catch.	It can catch errors that verification cannot catch.
Target is requirements specification, application and software architecture, high level, complete design, and database design, etc.	Target is an actual product-a unit, a module, a bent of integrated modules, and an effective final product.
Verification is done by QA team to ensure that the software is as per the specifications in the SRS document.	Validation is carried out with the involvement of the testing team
It generally comes first done before validation.	It generally follows after verification.
It is low-level exercise.	It is a High-Level Exercise.

13. What is a Baseline?

A baseline is a measurement that defines the completeness of a phase. After all activities associated with a particular phase are accomplished, the phase is complete and acts as a baseline for next phase.

For more details, please refer to the following article baseline.

Software Engineering Interview Questions for Intermediate

Here are the Top Software Engineering Interview Questions for **Intermediate:**

14. What is Cohesion and Coupling?

Cohesion indicates the relative functional capacity of the module. Aggregation modules need to interact less with other sections of other parts of the program to perform a single task. It can be said that only one coagulation module (ideally) needs to be run. Cohesion is a measurement of the functional strength of a module. A module with high cohesion and low coupling is functionally independent of other modules. Here, functional independence means that a cohesive module performs a single operation or function. The coupling means the overall association between the modules.

Coupling relies on the information delivered through the interface with the complexity of the interface between the modules in which the reference to the section or module was created. High coupling support Low coupling modules assume that there are virtually no other modules. It is exceptionally relevant when both modules exchange a lot of information. The level of coupling between two modules depends on the complexity of the interface.

For more details, please refer to the following article Coupling and cohesion.

15. What is the Agile software development model?

The
agile SDLC model is a combination of iterative and incremental process models with a focus on process adaptability and customer

involves cross-functional teams working simultaneously on various areas like planning, requirements analysis, design, coding, unit testing, and acceptance testing.

Advantages:

- Customer satisfaction by rapid, continuous delivery of useful software.
- Customers, developers, and testers constantly interact with each other.
- Close, daily cooperation between business people and developers.
- Continuous attention to technical excellence and good design.
- Regular adaptation to changing circumstances.
- Even late changes in requirements are welcomed.

For more details, please refer to the following article Software Engineering - Agile Development Models.

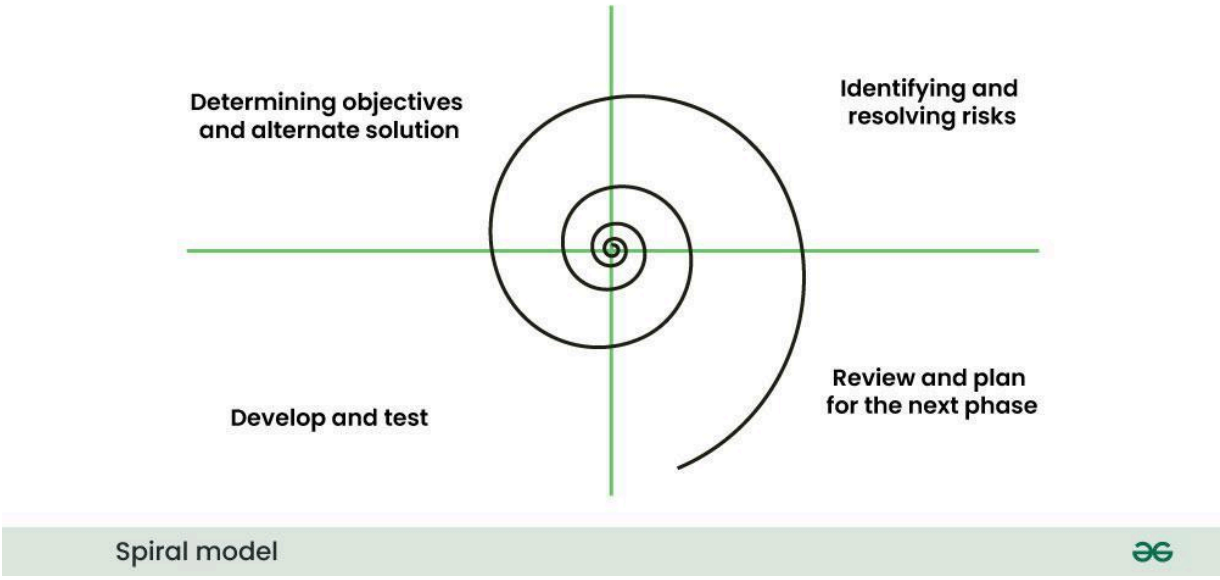
16. What is the Difference Between Quality Assurance and Quality Control?

Quality Assurance (QA)	Quality Control (QC)
It focuses on providing assurance that the quality requested will be achieved.	It focuses on fulfilling the quality requested.
It is the technique of managing quality.	It is the technique to verify quality.
It does not include the execution of the program.	It always includes the execution of the program.
It is a managerial tool.	It is a corrective tool.
It is process-oriented.	It is product-oriented.
The aim of quality assurance is to prevent defects.	The aim of quality control is to identify and improve the defects.
It is a preventive technique.	It is a corrective technique.

Quality Assurance (QA)	Quality Control (QC)
It is responsible for the full software development life cycle.	It is responsible for the software testing life cycle.
Example: Verification	Example: Validation

17. What is the Spiral Model, and its Disadvantages?

The Spiral Model is a **Software Development Life Cycle (SDLC)** model that provides a systematic and iterative approach to software development. In its diagrammatic representation, looks like a spiral with many loops. The exact number of loops of the spiral is unknown and can vary from project to project. Each loop of the spiral is called a **phase** of the software development process.



Spiral Model

Following are the disadvantages of spiral model:

- Can be a costly model to use.
- Risk analysis requires highly specific expertise.
- The project's success is highly dependent on the risk analysis phase.
- Doesn't work well for smaller projects

For more details, please refer to the following article Software Engineering - Spiral Model.

IBM first proposed the Rapid Application Development or RAD Model in the 1980s. The RAD model is a type of incremental process model in which there is a concise development cycle. The RAD model is used when the requirements are fully understood and the component-based construction approach is adopted.

Following are the limitations of RAD Model:

- For large but scalable projects RAD requires sufficient human resources.
- Projects fail if developers and customers are not committed in a much-shortened time frame.
- Problematic if a system cannot be modularized

For more details, please refer to the following article Software Engineering - Rapid Application Development Model (RAD).

19. What is Regression Testing?

Regression testing is defined as a type of software testing that is used to confirm that recent changes to the program or code have not adversely affected existing functionality. Regression testing is just a selection of all or part of the test cases that have been run. These test cases are rerun to ensure that the existing functions work correctly. This test is performed to ensure that new code changes do not have side effects on existing functions. Ensures that after the last code changes are completed, the above code is still valid.

*For more details, please refer to the following article **regression testing**.*

20. What are CASE Tools?

CASE stands for Computer-Aided Software Engineering. CASE tools are a set of automated software application programs, which are used to support, accelerate and smoothen the SDLC activities. It is a software package that helps with the design and deployment of information systems. It can record a database design and be quite useful in ensuring design consistency.

21. What is Physical and Logical DFD (Data Flow Diagram) ?

Physical DFD and Logical DFD both are the types of DFD (Data Flow

Physical DFD focuses on how the system is implemented. The next diagram to draw after creating a logical DFD is physical DFD. It explains the best method to implement the business activities of the system. Moreover, it involves the physical implementation of devices and files required for the business processes. In other words, physical DFD contains the implantation-related details such as hardware, people, and other external components required to run the business processes.

Logical data flow diagram mainly focuses on the system process. It illustrates how data flows in the system. In the Logical Data Flow Diagram (DFD), we focus on the high-level processes and data flow without delving into the specific implementation details. Logical DFD is used in various organizations for the smooth running of system. Like in a Banking software system, it is used to describe how data is moved from one entity to another.

22. What is Software Re-engineering?

Software re-engineering is the process of scanning, modifying, and reconfiguring a system in a new way. The principle of reengineering applied to the software development process is called software reengineering. It has a positive impact on software cost, quality, customer service, and shipping speed. Software reengineering improves software to create it more efficiently and effectively.

For more details please refer to What Is Software Re-Engineering?

23. What is Reverse Engineering?

Software Reverse Engineering is a process of recovering the design, requirement specifications, and functions of a product from an analysis of its code. It builds a program database and generates information from this. The purpose of reverse engineering is to facilitate maintenance work by improving the understandability of a system and producing the necessary documents for a legacy system.

Reverse Engineering Goals:

- Cope with Complexity.
- Recover lost information.

- Facilitate Reuse.

For more details, please refer to the following article Software Engineering - Reverse Engineering.

24. What are Software Project Estimation Techniques Available?

There are some software project estimation techniques available:

- PERT
- WBS
- Delphi method
- User case point

*For more details, please refer to the following article **Software Project Estimation Techniques**.*

25. How to Measure the Complexity of Software?

To measure the complexity of software there are some methods in software engineering:

- Line of codes
- Cyclomatic complexity
- Class coupling
- Depth of inheritance

For more details, please refer to the following article complexity of software.

26. Mentions Some Software Analysis and Design Tools?

Following are some software analysis and design tools:

- Data Flow Diagrams
- Structured Charts
- Structured English
- Data Dictionary
- Hierarchical Input Process Output diagrams

27. What is the Name of Various CASE Tools?

- Requirement Analysis Tool
- Structure Analysis Tool
- Software Design Tool
- Code Generation Tool
- Test Case Generation Tool
- Document Production Tool
- Reverse Engineering Tool

For more details, please refer to the following article Computer-Aided Software Engineering(CASE).

28. What is SRS?

Software Requirement Specification (SRS) Format is a complete specification and description of requirements of the software that needs to be fulfilled for successful development of software system. These requirements can be functional as well as non-requirements depending upon the type of requirement. The interaction between different customers and contractors is done because it is necessary to fully understand the needs of customers.

For more details please refer software requirement specification format article.

29. What is level-0 DFD?

The highest abstraction level is called Level 0 of DFD. It is also called context-level DFD. It portrays the entire information system as one diagram.

For more details, please refer to the following article DFD.

30. What is a Function Point?

Function point metrics provide a standardized method for measuring the various functions of a software application. Function point metrics, measure functionality from the user's point of view, that is, on the basis of what the user requests and receives in return.

31. What is the formula to Calculate the Cyclomatic Complexity?

The formula to calculate the cyclomatic complexity of a program is:

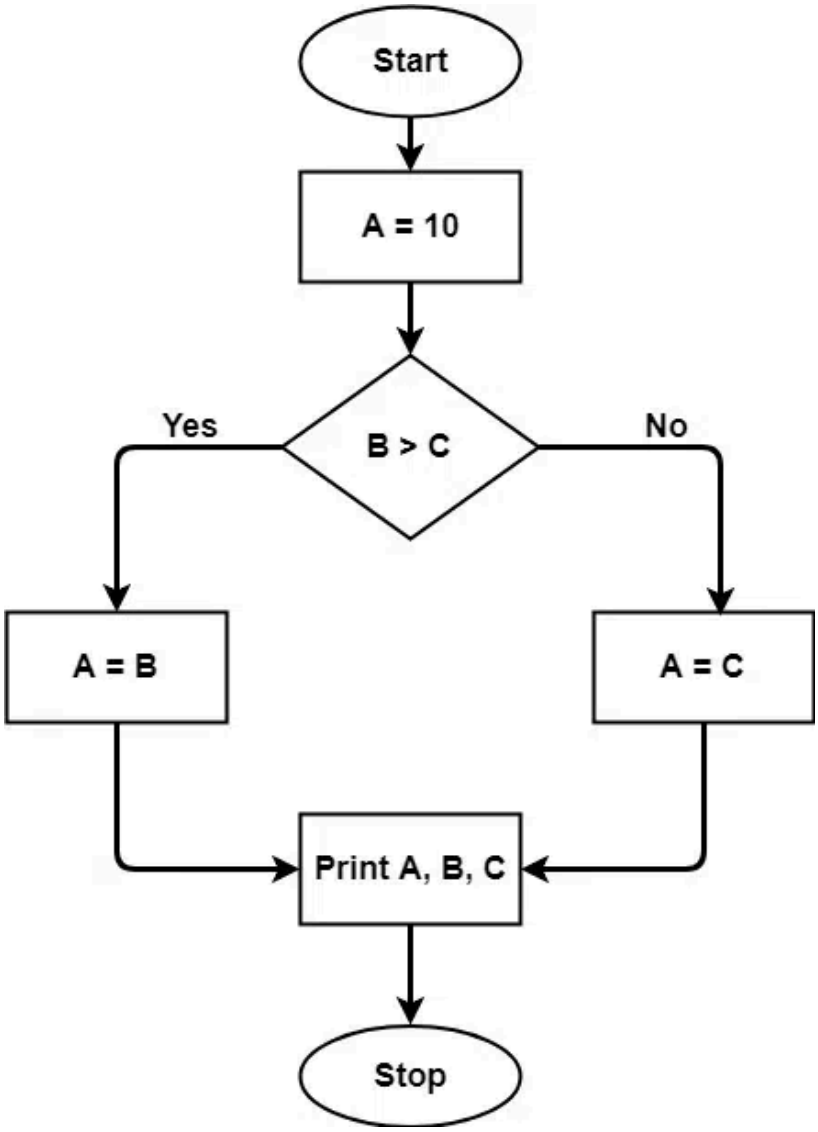
$$c = e - n + 2p \qquad \textit{where,}$$

1. *e = number of edges*
2. *n = number of vertices*
3. *p = predicates*

Example:

```
A = 10
   IF B > C THEN
      A = B
   ELSE
      A = C
   ENDIF
Print A
Print B
Print C
```

Control Flow Graph of the above code:



The cyclomatic complexity calculated for the above code will be from the control flow graph. The graph shows seven shapes(nodes), and seven lines(edges), hence cyclomatic complexity is 7-7+2 = 2.

32. What is the Cyclomatic Complexity of a Module that has 17 Edges and 13 Nodes?

The Cyclomatic complexity of a module that has seventeen edges and thirteen nodes = E – N + 2

E = Number of edges, N = Number of nodes
Cyclomatic complexity = 17 – 13 + 2 = 6

33. What is COCOMO Model?

A COCOMO model stands for Constructive Cost Model. As with all estimation models, it requires sizing information and accepts it in three forms:

- Function points
- Lines of source code

For more details, please refer to the following article Software Engineering - COCOMO Model.

34. Define an Estimation of Software Development Effort for Organic Software in the basic COCOMO Model?

Estimation of software development effort for organic software in the basic COCOMO model is defined as

```
Organic: Effort = 2.4(KLOC) 1.05 PM
```

Software Engineering Interview Questions for Advance

Here are the Top Software Engineering Interview Questions for **Advance:**

35. What Activities Come Under the Umbrella Activities?

The activities of the software engineering process framework are complemented by a variety of higher-level activities. Umbrella activities typically apply to the entire software project and help the software team manage and control progress, quality, changes, and risks. Common top activities include Software Project Tracking and Control Risk Management, Software Quality Assurance Technical Review Measurement Software Configuration Management Reusability Management Work Product Preparation and Production, etc.

For more details, please refer to the following article Umbrella activities in Software Engineering.

36. Which SDLC Model is the Best?

The selection of the best SDLC model is a strategic decision that requires a thorough understanding of the project's requirements, constraints, and goals. While each model has its strengths and weaknesses, the key is to align the chosen model with the specific characteristics of the project. Being flexible, adaptable, and communicating well are crucial in dealing with the

the end, the best way to develop software is the one that suits the project's needs and situation the most.

> *For more details, please refer to the following article Which SDLC Model is Best and Why ?*

37. What is the Black Hole Concept in DFD?

A block hole concept in the data flow diagram can be defined as "A processing step may have input flows but no output flows". In a black hole, data can only store inbound flows.

38. Which of the Testing is Used for Fault Simulation?

With increased expectations for software component quality and the complexity of components, software developers are expected to perform effective testing. In today's scenario, mutation testing has been used as a fault injection technique to measure test adequacy. Mutation Testing adopts "fault simulation mode".

39. Which Model is Used to Check Software Reliability?

A Rayleigh model is used to check software reliability. The Rayleigh model is a parametric model in the sense that it is based on a specific statistical distribution. When the parameters of the statistical distribution are estimated based on the data from a software project, projections about the defect rate of the project can be made based on the model.

40. What is the Difference between Risk and Uncertainty?

- Risk is able to be measured while uncertainty is not able to be measured.
- Risk can be calculated while uncertainty can never be counted.
- You are capable of make earlier plans in order to avoid risk. It is impossible to make prior plans for the uncertainty.
- Certain sorts of empirical observations can help to understand the risk but on the other hand, the uncertainty can never be based on empirical observations.
- After making efforts, the risk is able to be converted into certainty. On the

- After making an estimate of the risk factor, a decision can be made but as the calculation of the uncertainty is not possible, hence no decision can be made.

41. What is CMM?

To determine an organization's current state of process maturity, the SEI uses an assessment that results in a five-point grading scheme. The grading scheme determines compliance with a capability maturity model (CMM) that defines key activities required at different levels of process maturity. The SEI approach provides a measure of the global effectiveness of a company's software engineering practices and establishes five process maturity levels that are defined in the following manner:

- Level 1: Initial
- Level 2: Repeatable
- Level 3: Defined
- Level 4: Managed
- Level 5: Optimizing

For more details, please refer to the following article CMM.

43. A software does not wear out in the traditional sense of the term, but the software does tend to deteriorate as it evolves, why?

The software does not wear out in the traditional sense of the term, but the software does tend to deteriorate as it evolves because Multiple change requests introduce errors in component interactions. Unlike hardware, software doesn't physically wear out. However, it tends to deteriorate as it evolves because every time new features or updates are added, there's a chance of introducing new bugs or compatibility issues. Over time, multiple changes can cause different parts of the software to interact in unexpected ways, making it harder to maintain. If these issues aren't managed properly, the software becomes more complex and prone to failure, which affects its performance and reliability.

44. What are the elements to be considered in the System Model

The type and size of the software, the experience of use for reference to predecessors, difficulty level to obtain users' needs, development techniques and tools, the situation of the development team, development risks, the software development methods should be kept in mind. It is an important prerequisite to ensure the success of software development that designing a reasonable and suitable software development plan.

45. Define Adaptive Maintenance?

Adaptive maintenance defines as modifications and updating when the customers need the product to run on new platforms, on new operating systems, or when they need the product to interface with new hardware and software.

46. Define the term WBS?

The full form of WBS is Work Breakdown Structure. Its **Work Breakdown Structure** includes dividing a large and complex project into simpler, manageable, and independent tasks. For constructing a work breakdown structure, each node is recursively decomposed into smaller sub-activities, until at the leaf level, the activities become undividable and independent. A WBS works on a top-down approach.

For more detail please refer Work breakdown structure article.

47. How to Find the Size of a Software Product?

Estimation of the size of the software is an essential part of Software Project Management. It helps the project manager to further predict the effort and time which will be needed to build the project. Various measures are used in project size estimation. Some of these are:

- Lines of Code
- Number of entities in ER diagram
- Total number of processes in detailed data flow diagram
- Function points

48. What is Concurrency, and How to Achieve it ?

Concurrency refers to a system's ability to execute numerous tasks or processes at the same time, ostensibly concurrently. It is a wider concept that includes the idea of completing many jobs in concurrent intervals. In a concurrent system, tasks can begin, run, and finish in overlapping time frames, increasing overall system efficiency and responsiveness. There are many programming language available that support concurrency using unthreading example Java, C++ etc.

49. Why is Modularization important in Software Engineering?

Modular programming makes your code easier to read by dividing it into functions that only deal with one part of the overall functionality. When compared to monolithic code, it can make your files significantly smaller and easier to read.

50. Which Process Model Removes Defects before Software get into trouble?

Clean room software engineering is a software development approach to producing quality software. In clean room software engineering, an efficient and good quality software product is delivered to the client as QA (Quality Assurance) is performed each and every phase of software development.

51. Difference between an EXE and DLL?

DLL refers to Dynamic Link Library, and EXE is an executable. An EXE assembly can execute in its own address space, whereas a DLL cannot. It does not have its own address space, therefore it must run within a host and requires a consumer to invoke it.

Conclusion

Preparing for a **Software Engineering Interview** means having a solid foundation of both the basics and more advanced topics, like **SDLC models**, **testing methods**, and **design principles**. By mastering these areas, practicing problem-solving, and staying up-to-date with new technologies, You will be ready to clear any interview confidently with the well knowledge.

Before going to learn Node or ExpressJS it's better to learn the important JavaScript concepts that will be used in the backend.

1. Explain equality in JavaScript

In JavaScript, the `equality` operator is used for equality comparison. It compares two values and returns `true` if they are equal, and `false` otherwise. However, it performs type coercion, which means it converts the operands to the same type before making the comparison.

```
console.log(5 == '5'); // true, because '5' is conver
comparison
console.log(5 == 5);    // true, both values are of the same typ
console.log(5 == 6);    // false, the values are not equal
```

Output

```
true
true
false
```

2. Explain Function.prototype.bind.

In JavaScript, `Function.prototype.bind()` is a method that allows We can create a new function with a specified `this` value and optionally some initial arguments.

Syntax:

```
const newFunction = oldFunction.bind(thisArg, arg1, ag2, ...,
argN)
```

3. Explain the difference between Objects. freeze() vs const

- `Object.freeze()` is a method provided by JavaScript that freezes an object, making it immutable. This means that after calling `Object.freeze()` on an object, We cannot add, delete, or modify any of its properties.
- Even attempts to modify the properties of a frozen object will fail silently in non-strict mode and throw an error in strict mode.
- `Object.freeze()` operates on the object itself, making the object and its properties immutable.

Const

- `const` is a keyword in JavaScript used to declare constants. When We declare a variable using `const`, We cannot reassign it to a different value. However, this does not make the object itself immutable.
- If the variable is an object, its properties can still be modified or reassigned, but We cannot assign a new object to the variable.
- In other words, `const` ensures that the variable reference cannot change, but it does not ensure immutability of the object itself.

4. What are IIFEs (Immediately Invoked Function Expressions)?

IIFEs stands for Immediately Invoked Function Expressions. JavaScript functions that are executed immediately after they are defined. They are commonly used to create a new scope and encapsulate code, preventing variable declarations from polluting the global scope.

Syntax:

```
(function (){
// Function Logic Here.
})();
```

5. Can We describe the main difference between a .forEach

Here is the difference between forEach loop and map() loop

`.forEach()`:

- The `.forEach()` method is used to iterate over an array and execute a provided function once for each array element.
- It does not mutate the original array and does not return a new array.
- The callback function passed to `.forEach()` can perform actions on each element of the array, but it does not produce a new array.

`.map()`:

- The `.map()` method is used to iterate over an array and transform each element using a provided function.
- It returns a new array with the results of calling the provided function on each element of the original array.
- The callback function passed to `.map()` should return a new value for each element, which will be included in the new array.

6. What is a cookie? How can We create, read and clear cookies using Javascript?

A cookie is an important tool as it allows We to store the user information as a name-value pair separated by a semi-colon in a string format. If we save a cookie in our browser then we can log in directly to the browser because it saves the user information.

- **Create cookies**: We can apply various operations on cookie-like create, delete, read, add an expiry date to it so that users can never be logged in after a specific time. A cookie is created by the **document.cookie** keyword.
- **Read cookies**: This function retrieves the cookie data stored in the browser. The cookie string is automatically encoded while sending it from the server to the browser.

the cookies when the date and time exceed the expiration date (and time).

7. What are Closures in JavaScript?

JavaScript closure is a feature that allows inner functions to access the outer scope of a function. Closure helps in binding a function to its outer boundary and is created automatically whenever a function is created.

```
function foo(outer_arg) {

    function inner(inner_arg) {
        return outer_arg + inner_arg;
    }
    return inner;
}
let get_func_inner = foo(5);

console.log(get_func_inner(4));
console.log(get_func_inner(3));
```

Output

```
9
8
```

The `foo` function returns an inner function that adds its argument to the outer function's argument. It creates a closure, preserving the outer function's state.

8. What are the arrow functions in JavaScript?

Arrow function {()=>} is concise way of writing JavaScript functions in shorter way. **Arrow functions** were introduced in the ES6 version. They make our code more structured and readable. Mostly in

```
const gfg = () => {
    console.log( "Hi Geek!" );
}
gfg() // output will be Hi Geek!
```

9. What are Imports and Exports in JavaScript?

Exports:

- Exports are used to expose functionality from a module to other parts of the program.
- We can export variables, functions, classes, or any other JavaScript entity by using the `export` keyword.
- There are different ways to export:
  - Default Export: `export default myFunction;`
  - Named Export: `export const myVariable = 10;`
  - Named Exports from Expressions: `export { myFunction };`
- We can have multiple exports in a single module.

Imports:

- Imports are used to bring functionality from other modules into the current module.
- We can import exported entities using the `import` keyword.
- We can import default exports like this: `import myFunction from './module';`
- We can import named exports like this: `import { myVariable } from './module';`
- We can also import all named exports using the `* as` syntax: `import * as module from './module';`

10. What are the various Operators in Javascript?

JavaScript **operators** operate the operands, these are symbols that are

performing specific mathematical and logical computations on operands.

NodeJS Backend Developer Interview Questions

11. What is NodeJS and how it works?

NodeJS is a JavaScript runtime environment that allows developers to run JavaScript code outside of a web browser. It uses the V8 JavaScript engine, which is the same engine used by Google Chrome, to execute JavaScript code on the server-side.

- NodeJS works on an event-driven, non-blocking I/O model, enabling efficient and scalable server-side applications.
- It uses asynchronous programming to handle multiple requests simultaneously, making it ideal for real-time web apps, APIs, and microservices.
- With a vast library ecosystem, NodeJS extends its capabilities for various use cases.

12. How do we manage packages in a NodeJS project?

Modules are the blocks of reusable code consisting of Javascript functions and objects which communicate with external applications based on functionality. A **package** is often confused with modules, but it is simply a collection of modules (libraries).

13. How can we use async await in NodeJS?

To use async await in NodeJS:

- Define an asynchronous function with the `async` keyword.
- Use the `await` keyword within the function to pause execution until promises are resolved.
- Handle errors using `try/catch` blocks.

```
async function fun1(req, res){
  let response = await request.get('http://localhost:3000');
    if (response.err) { console.log('error');}
    else { console.log('fetched response');
}
```

14. What is NodeJS streams?

Streams are one of the fundamental concepts of NodeJS. Streams are a type of data-handling methods and are used to read or write input into output sequentially. Streams are used to handle reading/writing files or exchanging information in an efficient way.

Accessing Streams:

```
const fs = require("fs");

const readStream = fs.createReadStream("source.txt");
const writeStream = fs.createWriteStream("destination.txt");

readStream.pipe(writeStream)
  .on("error", console.log);

writeStream.on("finish", () =>
  console.log("Data successfully copied!")
);
```

15. Describe the exit codes of NodeJS?

NodeJS is a cross-platform, open-source back-end JavaScript runtime environment that uses the V8 engine to execute JavaScript code outside of a web browser. Exit codes in NodeJS are a specific group of codes that finish off processes, which can include global objects as well.

16. Explain the concept of stub in NodeJS?

- Stubs can be either anonymous.
- Stubs can be wrapped into existing functions. When we wrap a stub into the existing function the original function is not called.

17. How can we enhance the performance of NodeJS?

Clustering is the process through which we can use multiple cores of our central processing unit at the same time with the help of NodeJS, which helps to increase the performance of the software and also reduces its time load.

We can install cluster modules through the given command.

```
npm i cluster
```

18. What is WASI and why is it being introduced?

WASI provides a standard way to talk to the outside world like operating system so that so that it can ask to read the file or perform the other operations .

- **WASI (WebAssembly System Interface)** acts as a bridge between WebAssembly and the operating system.
- **WebAssembly** translates languages like C++ or Rust into a format browsers understand.
- **Browsers only understand JavaScript**, so WebAssembly enables execution of other languages.
- **WASI provides a standard way** for WebAssembly to interact with the OS.
- It allows WebAssembly to **perform system operations** like file reading and writing.

19. How to measure the duration of async operations?

code then the code after that is executed and does not wait for this asynchronous operation to complete.

Syntax

```
const calcTime = async () => {
    const start = Date.now();
    await someAsyncOperation();
    const end = Date.now()
    const duration = end - start;
}
```

20. What is a thread pool and which library handles it in NodeJS?

A thread pool is a collection of worker threads that are used to execute asynchronous tasks concurrently in a multithreaded environment.

In NodeJS, the `libuv` library handles the thread pool.

- Thread pool is a collection of worker threads for concurrent task execution.
- Prevents new thread creation for each task, improving efficiency.
- NodeJS uses libuv to manage the thread pool.
- Libuv handles async I/O, including file system, network, and timers.

ExpressJs Backend Developer Interview Questions

21. How does ExpressJS handle middleware?

ExpressJS is a routing and Middleware framework for handling the different routing of the webpage and it works between the request and response cycle. Middleware gets executed after the server receives the request and before the controller actions send the

```
app.get(path, (req, res, next) => {}, (req, res) => {})
```

22. How does ExpressJS handle request and response objects?

In ExpressJS, request and response objects are fundamental to handling HTTP requests and generating HTTP responses. Here's how ExpressJS handles these objects:

1. **Request Object (req):** Represents the client's HTTP request, containing method, URL, headers, query params, and body. Express passes it to route handlers and middleware.
2. **Response Object (res):** Represents the server's response, allowing data sending, status control, and header setting. Supports methods like res.json(), res.send(), res.redirect(), and res.setHeader().

23. How can we create a Rest API route in ExpressJS?

NodeJS server code sets up a RESTful API for managing data. It provides endpoints for performing CRUD (Create, Read, Update, Delete) operations on a collection of records. The server uses the ExpressJS framework to handle HTTP requests.

24. What is the difference between a traditional server and an ExpressJS server?

Traditional server	ExpressJS server
Built using frameworks like Spring (Java), Django (Python), Ruby on Rails, ASP.NET, etc.	Lightweight and flexible NodeJS framework for web apps and APIs.
It Supports both synchronous and asynchronous programming	Built on NodeJS's event-driven, non-blocking I/O model for

Traditional server	ExpressJS server
It Comes with **built-in middleware and routing systems** in most frameworks.	Provides middleware and routing, but with a minimalist and customizable approach.
Different frameworks have varying community sizes and ecosystems.	Large and active community with a rich ecosystem of plugins and extensions.

25. What is Token-based Authentication? What is JWT?

Token-Based Authentication is a security mechanism where a user logs in and receives a token, which is then used for subsequent requests instead of credentials. This approach enhances security and scalability, especially in stateless applications like REST APIs.

A JSON web token(JWT) is a compact, self-contained token format used for securely transmitting information between parties. It consists of three parts:

- **Header:** Contains metadata, such as the token type and signing algorithm.
- **Payload:** Holds the claims (user data or permissions).
- **Signature:** Ensures integrity and authenticity by verifying the token with a secret key.

26. What is Pub-Sub architecture?

Publisher Subscriber basically known as Pub-Sub is an asynchronous message-passing system that solves the drawback above. The sender is called the publisher whereas the receiver is called the subscriber. The main advantage of pub-sub is that it decouples the subsystem

POST parameter can be received from a form using express.urlencoded() middleware and the req.body Object. The express.urlencoded() middleware helps to parse the data that is coming from the client-side.

```
const express = require('express');
const app = express();

app.use(express.urlencoded({ extended: true }));

app.post('/submit', (req, res) =&gt; {
    const { name, email } = req.body;

    console.log('Received form submission:');
    console.log('Name:', name);
    console.log('Email:', email);

    res.send('Form submitted successfully');
});
const port = process.env.PORT || 3000;
app.listen(port, () =&gt; {
    console.log(`Server is running on port ${port}`);
});
```

In this code

- app.use(express.urlencoded({ extended: true })) enables parsing of application/x-www-form-urlencoded request bodies.
- A POST route /submit handles form submissions.
- **req.body** extracts name and email via middleware.
- Data is processed (logged to console).
- Response sent using `res.send()`.

28. What do We understand by Scaffolding in ExpressJS?

Scaffolding is creating the skeleton structure of application. It allows users to create own public directories, routes, views etc. Once the structure for app is built, user can start building it.

Syntax

```
npm install express --save
```

Error handling in ExpressJS refers to the process of capturing and responding to errors that occur during the execution of an Express application.

In Express, We can handle errors using middleware functions, which are functions that have access to the request and response objects, as well as the next middleware function in the application's request-response cycle.

30. How to do Templating using ExpressJS in NodeJS?

A template engine basically helps us to use the static template files with minimal code. At runtime, the template engine replaces all the variables with actual values at the client-side.

To use templating with ExpressJS in NodeJS:

1. Install a template engine like EJS (`npm install ejs`).
2. Set up Express to use the template engine (`app.set('view engine', 'ejs')`).
3. Create EJS templates in the `views` directory.
4. Render EJS templates in Express routes using `res.render()`.
5. Pass dynamic data to the templates.
6. Start the Express server.

SQL Backend Developer Interview Questions

31. What is the difference between LEFT JOIN with WHERE clause & LEFT JOIN?

LEFT JOIN with WHERE Clause:

```
SELECT *
FROM table1
LEFT JOIN table2 ON table1.column = table2.column
```

- This query performs a LEFT JOIN between table1 and table2 based on the specified column.
- The WHERE clause filters the results to only include rows where there is no match in table2 (i.e., table2.column IS NULL).
- It effectively retrieves records from table1 and the matching records from table2, where no match is found, the columns from table2 will be NUL.

Regular LEFT JOIN:

```
SELECT *
FROM table1
LEFT JOIN table2 ON table1.column = table2.column
WHERE table2.column IS NULL;
```

- Retrieves records from table1 where no match is found in table2.
- The WHERE table2.column IS NULL filters out matching rows.
- Used to find unmatched records from table1.

32. How do We prevent SQL Server from giving We informational messages during and after a SQL statement execution?

In SQL Server, informational messages, often called "info" or "print" messages, can be generated during and after the execution of a SQL statement. These messages might include details such as the number of rows affected or certain warnings. If We want to suppress these informational messages, We can use the SET command with the NOCOUNT option.

```
SET NOCOUNT ON;
-- Wer SQL statements go here
SET NOCOUNT OFF;
```

- **SET NOCOUNT OFF;**: This setting turns the informational messages back on.

33. What is Cursor? How to use a Cursor?

Cursor is a Temporary Memory or Temporary Work Station. It is Allocated by Database Server at the Time of Performing DML(Data Manipulation Language) operations on the Table by the User. Cursors are used to store Database Tables.

There are two types of Cursors:

1. **Implicit Cursors:** Implicit Cursors are also known as Default Cursors of SQL SERVER. These Cursors are allocated by SQL SERVER when the user performs DML operations.
2. **Explicit Cursors:** Explicit Cursors are Created by Users whenever the user requires them. Explicit Cursors are used for Fetching data from Table in Row-By-Row Manner.

34. How SQL Server executes a statement with nested subqueries?

When SQL Server processes a statement with nested subqueries execution process involves evaluating each subquery from the innermost to the outermost level. The general steps for executing a statement with nested subqueries are as follows:

- **Parsing & Compilation:** SQL Server parses the query and creates an execution plan.
- **Innermost Subquery Execution:** The innermost subquery runs first, producing a value or set of values.
- **Intermediate Storage:** If needed, results are stored in temporary structures.
- **Propagation:** These results are passed to the next query level.
- **Higher-Level Execution:** This process continues for higher-level

- **Result Combination:** Data from all levels is combined to produce the final output.
- **Query Optimization:** SQL Server optimizes execution by reordering joins and selecting efficient indexes.

35. What is a deadlock and what is a live lock?

Deadlock: Occurs when two or more transactions wait indefinitely for each other to release a resource, causing a standstill.

How to Fix Deadlocks:

1. **Prevention:**

- Allocate resources carefully to avoid conflicts.
- Follow a fixed order when requesting resources.

2. **Detection & Resolution:**

- Use timeouts or deadlock detection mechanisms.
- Rollback one transaction (victim) to allow others to proceed.

3. Transaction Timeout:

- Set a time limit for transactions; rollback if it exceeds the limit.

4. Lock Hierarchy:

- Always acquire locks in a structured order to prevent circular waits.

5. Avoidance Algorithms:

- Predict and prevent deadlocks before allocating resources.

37. Where is the MyISAM table stored?

Stored as Files: Each MyISAM table has three files:

- .frm – Table structure.

Location:

- Linux: /var/lib/mysql/
- Windows: C:\ProgramData\MySQL\MySQL Server X.Y\data\

Locking: Uses table-level locking, which can cause contention.

Optimization: OPTIMIZE TABLE improves performance.

Consideration: InnoDB is preferred for transactions and better concurrency.

MongoDB Backend Developer Interview Questions

38. What is BSON in MongoDB?

BSON stands for Binary JSON. It is a binary file format that is used to store serialized JSON documents in a binary-encoded format.

The MongoDB database had several scalar data formats that were of special interest only for MongoDB, hence they developed the BSON data format to be used while transferring files over the network.

39. What are Replication and Sharding in MongoDB?

Replication - is the method of duplication of data across multiple servers. For example, we have an application and it reads and writes data to a database and says this server A has a name and balance which will be copied/replicate to two other servers in two different locations.

Sharding - is a method for allocating data across multiple machines. MongoDB used sharding to help deployment with very big data sets and large throughput the operation. By sharding, We combine more devices to carry data extension and the needs of read and write operations.

MongoDB provides We read operations to retrieve embedded/nested documents from the collection or query a collection for a embedded/nested document. We can perform read operations using the db.collection.find() method.

41. Can We create an index on an array field in MongoDB? If yes, what happens in this case?

Yes, We can create an index on a field containing an array value to improve performance for queries on that field. When We create an index on a field containing an array value, MongoDB stores that index as a multikey index.

To create an index, use the **db.collection.createIndex() method**.

```
db.<collection>.createIndex( { <field>: <sortOrder> } )
```

The example uses a students collection that contains these documents:

```
db.students.insertMany([
    {
        "name": "Andre Robinson",
        "test_scores": [88, 97]
    },
    {
        "name": "Wei Zhang",
        "test_scores": [62, 73]
    },
    {
        "name": "Jacob Meyer",
        "test_scores": [92, 89]
    }
])
```

Procedure: The following operation creates an ascending multikey index on the test_scores field of the students collection:

```
db.students.createIndex( { test_scores: 1 } )
```

Shard key, which is a unique identifier that is used to map the data to its corresponding shard. When a query is received, the system uses the shard key to determine which shard contains the required data and then sends the query to the appropriate server or node.

43. What is ObjectId in MongoDB?

Every document in the collection has an "_id" field that is used to uniquely identify the document in a particular collection it acts as the primary key for the documents in the collection. "_id" field can be used in any format and the default format is **ObjectId** of the document.

```
ObjectId(<hexadecimal>)
```

44. What is a Covered Query in MongoDB?

A covered query in MongoDB is a type of query where all the fields needed for the query are covered by an index. For a query to be considered covered the following conditions must be met:

- **Projection:** The query must include a projection that only selects fields covered by the index. A projection specifies which fields to include or exclude in the query results.
- **Index:** The fields specified in the query's filter criteria must be part of an index. The fields specified in the projection must be part of the same index.
- **No Additional Document Fields:** The query should not include additional fields or expressions that are not covered by the index. If additional fields are needed, the index might not cover the entire query.

```
{ "_id": 1, "name": "Alice", "age": 30, "department": "HR" }
{ "_id": 2, "name": "Bob", "age": 35, "department": "IT" }
```

```
db.employees.createIndex({ "department": 1 });
```

Now, if We perform a query like this:

```
db.employees.find({ "department": "HR" }, { "_id": 0, "name": 1
});
```

In this query:

- The filter criteria ({ "department": "HR" }) matches the indexed field.
- The projection ({ "_id": 0, "name": 1 }) includes only the "name" field, which is also part of the index.

45. How to Create Relationship in MongoDB?

In MongoDB, a relationship represents how different types of documents are logically related to each other. Relationships like one-to-one, one-to-many, etc., can be represented by using two different models:

1. Embedded document model
2. Reference model

46. What is CAP theorem?

The **CAP theorem**, originally introduced as the CAP principle, can be used to explain some of the competing requirements in a distributed system with replication. It is a tool used to make system designers aware of the trade-offs while designing networked shared-data systems.

47. What is Indexing in MongoDB?

MongoDB uses indexing in order to make the query processing more efficient. If there is no indexing, then the MongoDB must scan every

information related to the documents such that it becomes easy for MongoDB to find the right data file.

Django Backend Developer Interview Questions

48. Explain Django Architecture?

Django is a Python-based web framework which allows We to quickly create web application without all of the installation or dependency problems that We normally will find with other frameworks. Django is based on MVT (Model-View-Template) architecture. MVT is a software design pattern for developing a web application.

49.What are templates in Django or Django template language?

Templates are the third and most important part of Django's MVT Structure. A template in Django is basically written in HTML, CSS, and Javascript in a .html file. Django framework efficiently handles and generates dynamic HTML web pages that are visible to the end-user.

50. What is Django ORM?

Django lets us interact with its database models, i.e. add, delete, modify, and query objects, using a database-abstraction API called ORM(Object Relational Mapper). Django's Object-Relational Mapping (ORM) system, a fundamental component that bridges the gap between the database and the application's code.

51. How to get a particular item in the Model?

In Django, We can retrieve a particular item (a specific record or instance) from a model using the model's manager and a query.

- **Import the Model:** Make sure to import Wer model in the Python file where We need to retrieve the item.

```
from Wer_app.models import WerModel
```

- **Use the Model's Manager:** Every Django model comes with a default manager called objects. We can use this manager to perform queries on the model.
- **Perform the Query:** Use a query to filter the items based on the criteria We want. For example, if We want to retrieve an item by its primary key (id field), We can use the get method.

52. Difference between select related and prefetch related?

In Django, select_related and prefetch_related are designed to stop the deluge of database queries that are caused by accessing related objects.

- **select_related()** “follows” foreign-key relationships, selecting additional related-object data when it executes its query.
- **prefetch_related()** does a separate lookup for each relationship and does the “joining” in Python.

53. How can We combine multiple QuerySets in a View?

A QuerySet is a collection of database queries to retrieve data from Wer database. It represents a set of records from a database table or a result of a database query. Query sets are lazy, meaning they are not evaluated until We explicitly request the data, which makes them highly efficient.

54. What is Django Field Choices?

Django Field Choices. According to documentation Field Choices are a

are given, they're enforced by model validation and the default form widget will be a select box with these choices instead of the standard text field.

55. What are Django URLs?

In Django, views are Python functions which take a URL request as parameter and return an HTTP response or throw an exception like 404. Each view needs to be mapped to a corresponding URL pattern. This is done via a Python module called URLConf (URL configuration).

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('books.urls')),
]
```

56. What are the views of Django?

Django Views are one of the vital participants of MVT Structure of Django. As per Django Documentation, A view function is a Python function that takes a Web request and returns a Web response. This **response** can be the HTML contents of a Web page, or a redirect, or a 404 error, or an XML document, or an image, anything that a web browser can display.

57. What are the models in Django?

A **Django model** is the built-in feature that Django uses to create tables, their fields, and various constraints. In short, Django Models is the SQL Database one uses with Django. SQL (Structured Query Language) is complex and involves a lot of different queries for creating, deleting, updating, or any other stuff related to a database.

```
    title = models.CharField(max_length = 200)
    description = models.TextField()
```

PHP Backend Developer Interview Questions

58. How do We enable error reporting in PHP?

The **error_reporting()** function sets the error_reporting directive at runtime. PHP has many levels of errors, using this function sets that level for the duration (runtime) of Wer script. If the optional error_level is not set, **error_reporting()** will just return the current error reporting level.

59. How is the explode() function used?

The explode() function is an inbuilt function in PHP used to split a string into different strings. The explode() function splits a string based on a string delimiter, i.e. it splits the string wherever the delimiter character occurs. This function returns an array containing the strings formed by splitting the original string.

Syntax:

```
array explode(separator, OriginalString, NoOfElements)
```

60. How does the array walk function work in PHP?

The *array_walk()* function is an inbuilt function in PHP. The array_walk() function walks through the entire array regardless of pointer position and applies a callback function or user-defined function to every element of the array. The array element's keys and values are parameters in the callback function.

Syntax:

A web browser may be the client, and an application on a computer that hosts a website may be the server. A client (browser) submits an HTTP request to the server; then the server returns a response to the client. The response contains status information about the request and may also contain the requested content.

- **GET:** Requests data from a specified resource.
- **POST:** Submits data to be processed to a specified resource.

62. When would We use === instead of ==?

- **Operator:** This operator is used to check the given values are equal or not. If yes, it returns *true*, otherwise it returns *false*.

```
operand1 == operand2
```

- **=== Operator:** This operator is used to check the given values and its data type are equal or not. If yes, then it returns *true*, otherwise it returns *false*.

```
operand1 === operand2
```

63. What is MVC and what does each component do?

The **Model-View-Controller (MVC)** framework is an architectural/design pattern that separates an application into three main logical components **Model**, **View**, and **Controller**. Each architectural component is built to handle specific development aspects of an application. It isolates the business logic and presentation layer from each other.

64. What are the differences between die() and exit() functions in PHP?

event of a mistake. Use **exit()** when there is not an error and have to stop the execution.

```
exit("Message goes here");
or
exit();
```

PHP die() Function: In PHP, **die()** is the same as **exit()**. A program's result will be an empty screen. Use die() when there is an error and have to stop the execution.

```
die("Message goes here");
or
die();
```

65. What is stdClass in PHP?

The stdClass is the empty class in PHP which is used to cast other types to object. It is similar to Java or Python object. The stdClass is not the base class of the objects. If an object is converted to object, it is not modified. But, if object type is converted/type-casted an instance of stdClass is created, if it is not NULL. If it is NULL, the new instance will be empty.

66. How would We create a Singleton class using PHP?

When We don't want to have more than a single instance of a given class, then the **Singleton Design Pattern** is used and hence the name is **Singleton**.

- Singleton is the design patterns in PHP OOPs concept that is a special kind of class that can be instantiated only once.
- If the object of that class is already instantiated then, instead of creating a new one, it gets returned.

Spring Boot is built on the top of the spring and contains all the features of spring. And is becoming a favorite of developers these days because of its rapid production-ready environment which enables the developers to directly focus on the logic instead of struggling with the configuration and setup.

Java Spring Backend Developer Interview Questions

68. What Are the Benefits of Using Spring?

Here is the benefits of using spring:

1. **Modularity**: Lightweight and modular design.
2. **Inversion of Control (IoC)**: Manages object lifecycle and reduces coupling.
3. **Aspect-Oriented Programming (AOP)**: Modularizes cross-cutting concerns.
4. **Dependency Injection (DI)**: Promotes loose coupling and testability.
5. **Transaction Management**: Simplifies database transactions.
6. **Integration**: Seamlessly integrates with existing technologies.
7. **Enterprise Features**: Provides support for security, caching, messaging, etc.
8. **Testability and Maintainability**: Promotes unit testing and modular design.

69. What is Spring Boot Multi-Module Project.

A Spring Boot Multi-Module Project is a project structure where We organize Wer codebase into multiple modules, each representing a different functional or logical unit of Wer application. Spring Boot, a popular Java framework, provides support for creating and managing multi-module projects.

In a multi-module project, We typically have a parent module (also

70. What Is Dependency Injection?

Dependency Injection is the main functionality provided by Spring IOC(Inversion of Control). The Spring-Core module is responsible for injecting dependencies through either Constructor or Setter methods. The design principle of Inversion of Control emphasizes keeping the Java classes independent of each other and the container frees them from object creation and maintenance.

71. What Is the Difference Between BeanFactory and ApplicationContext?

Here is the differnce between BeanFactory and ApplicationContext

The BeanFactory Interface

This is the root interface for accessing a Spring bean container. It is the actual container that instantiates, configures, and manages a number of beans. These beans collaborate with one another and thus have dependencies between themselves.

```
ClassPathResource resource = new ClassPathResource("beans.xml");
XmlBeanFactory factory = new XmlBeanFactory(resource);
```

The ApplicationContext Interface

This interface is designed on top of the BeanFactory interface. The ApplicationContext interface is the advanced container that enhances BeanFactory functionality in a more framework-oriented style.

```
ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");
```

72. Describe the Spring bean lifecycle.

then dependencies are injected. And finally, the bean is destroyed when the spring container is closed.

73. How to Set Context Path in Spring Boot Application?

The context path is a prefix to the URL path used to identify and differentiate between different context(s). In Spring Boot, by default, the applications are accessed by context path **"/"**. That means we can access the application directly at ***http://localhost:PORT/***. For example

```
http://localhost:8080/
```

74. Explain the difference between @Before, @After, and @Around advice in AOP.

Here is the difference between @Before, @After, and @Around

- **@Around**: This is the most effective advice among all other advice. The first parameter is of type ProceedingJoinPoint. Code should contain proceed() on the ProceedingJoinPoint and it causes the underlying lines of code to execute.
- **@Before:** This advice will run as a first step if there is no @Around advice. If @Around is there, it will run after the beginning portion of @Around.
- **@After**: This advice will run as a step after @Before advice if there is no @Around advice. If @Around is there, it will run after the ending portion of @Around.
- **@AfterReturning**: This advice will run as a step after @After advice. Usually, this is the place , where we need to inform about the successful resultant of the method.

75. What is autowiring in spring?

The Spring container detects those dependencies specified in the

- An autowired application requires fewer lines of code comparatively but at the same time, it provides very little flexibility to the programmer.

76. What Is Spring Security?

Spring Security is a framework that allows a programmer to use JEE components to set security limitations on Spring-framework-based Web applications. In a nutshell it's a library that can be utilized and customized to suit the demands of the programmer.

77. What is Spring IOC Container?

Spring IoC Container is the core of Spring Framework. It creates the objects, configures and assembles their dependencies, manages their entire life cycle. The Container uses Dependency Injection(DI) to manage the components that make up the application.

API Backend Developer Interview Questions

78. What is an API (Application Programming Interface)?

API **is an abbreviation for Application Programming Interface** which is a collection of communication protocols and subroutines used by various programs to communicate between them. A programmer can make use of various API tools to make their program easier and simpler.

79. What is REST?

Representational **S**tate **T**ransfer (REST) is an architectural style that defines a set of constraints to be used for creating web services.

- **REST API** is a way of accessing web services in a simple and

80. What is a URI?

URI stands for Uniform Resource Identifier. It is a technical term that used for the names of all resources Connected to the World Wide Web. URIs established the protocols over the internet to conduct the connection between among resources.

81. What is RESTful?

RESTful web services are generally highly scalable, light, and maintainable and are used to create APIs for web-based applications. It exposes API from an application in a secure and stateless manner to the client. The protocol for REST is HTTP.

82. What are the REST API Architectural Constraints?

REST is a software architectural style that defines the set of rules to be used for creating web services. Web services which follow the REST architectural style are known as RESTful web services. There are six architectural constraints which makes any web service are listed below:

- Uniform Interface
- Stateless
- Cacheable
- Client-Server
- Layered System
- Code on Demand

83. What is the difference between the POST method and the PUT method?

Here is the difference between the Post and Put method :

stored under the supplied URI.

```
import requests
# Making a PUT request
r = requests.put('https://httpbin.org/put', data={'key':'value'})

#check status code for response received
# success code - 200
print(r)

# print content of request
print(r.content)
```

HTTP POST Request

HTTP POST is a request method supported by HTTP used by the World Wide Web. By design, the POST request method requests that a web server accepts the data enclosed in the body of the request message, most likely for storing it.

```
import requests
# Making a POST request
r = requests.post('https://httpbin.org/post', data={'key':'value'})

#check status code for response received
# success code - 200
print(r)

# print content of request
print(r.json())
```

84. What are HTTP Status Codes?

HTTP status codes are a set of standardized three-digit numbers used by web servers to communicate the outcome of an HTTP request made by a client. They provide information about whether the request was successful, encountered an error, or requires further action by the client.

- 1xx - Informational: Request received, continuing process.
- 2xx - Success: The action was successfully received, understood

- 4xx - Client Error: The request contains bad syntax or cannot be fulfilled.
- 5xx - Server Error: The server failed to fulfill an apparently valid request.

85. How do We keep REST APIs secure?

There are ***three types*** of security mechanism for an API –

- **HTTP Basic Authentication:** In this mechanism HTTP User Agent provides a Username and Password. Since this method depends only on HTTP Header and entire authentication data is transmitted on insecure lines
- **API Keys:** API Keys came into picture due to slow speed and highly vulnerable nature of HTTP Basic Authentication.
- **OAuth:** OAuth is not only a method of Authentication or Authorization, but it's also a mixture of both the methods. Whenever an API is called using OAuth credential, user logs into the system, generating a token.

86. What are cache-control headers?

The **Cache-Control header** is a general header, that specifies the caching policies of server responses as well as client requests. Basically, it gives information about the manner in which a particular resource is cached, location of the cached resource, and its maximum age attained before getting expired i.e. time to live.

Syntax:

```
Cache-Control: <directive> [, <directive>]*
```

87. What is API Integration?

Commonly Asked Data Structure Interview Questions

Last Updated : 23 Jul, 2025

To excel in a Data Structure interview, a strong grasp of fundamental concepts is crucial. Data structures provide efficient ways to store, organize, and manipulate data, making them essential for solving complex problems in software development.

Interviewers often test candidates on various data structures, their operations, and their real-world applications. Understanding the strengths and limitations of each data structure helps in selecting the right one for a given problem.

1. Array

- Efficient access and modification of elements using indices, ideal for tasks like searching and sorting.
- Operations like insertion, deletion, and traversal have varying time complexities, such as O(n) for deletion.
- A common interview focus includes handling dynamic arrays and solving problems like finding the maximum subarray.

Read more about Commonly Asked Data Structure Interview Questions on Array

2. Matrix

- Used for representing data in a 2D format, with efficient element

- Dynamic programming and graph-related problems often require optimized matrix manipulation.

Read more about Commonly Asked Data Structure Interview Questions on Matrix

3. Linked List

- A dynamic data structure with nodes connected by pointers (In C and C++) or references (in Java, Python and JavaScript), ideal for situations where elements are frequently added or removed.
- Common interview problems include reversing a list, detecting cycles, and merging sorted lists.
- Understanding recursion and pointer manipulation is key to solving problems efficiently.

Read more about Commonly Asked Data Structure Interview Questions on Linked List

4. Hashing

- Hashing maps data to a fixed-size table using a hash function, enabling fast lookups. It's commonly used in databases, caching, and cryptography.
- Interview questions often cover hash functions, collision handling (chaining, open addressing), and applications like anagram detection and LRU cache.
- The problems on hashing also involve set operations (union, intersection, subset), subarray sum (prefix sum and hashing used here) and frequency counting.

Read more about Commonly Asked Data Structure Interview Questions on Hashing

- Algorithms for locating an element in a dataset, such as linear search, binary search, and hash-based methods.
- Interview questions often involve binary search variations (e.g., searching in rotated sorted arrays) and searching algorithms in unsorted data.
- Understanding time complexities (O(log n) for binary search, O(n) for linear search) is key for optimizing solutions.

Read more about Commonly Asked Data Structure Interview Questions on Searching

6. Sorting

- Algorithms that arrange elements in a specific order, such as ascending or descending, crucial for optimizing search operations.
- Common algorithms include Quick Sort, Merge Sort, and Heap Sort, with interview questions focusing on their time and space complexities.
- Advanced sorting techniques like Counting Sort and Radix Sort are useful for specific types of input.

Read more about Commonly Asked Data Structure Interview Questions on Sorting

7. Strings

- Used to store and manipulate sequences of characters, essential for tasks like pattern matching and text processing.
- Interview questions often focus on string reversal, palindrome checks, and substring search (KMP, Rabin-Karp).
- Understanding string manipulation techniques is crucial for optimizing algorithms in real-world applications.

8. Stack

- Follows the Last In, First Out (LIFO) principle, useful for operations like parsing expressions and managing function calls in recursion.
- Common interview problems include balanced parentheses, next greater element (and its variations like previous greater, largest area in histogram, stock span and trapping rain water), and evaluating postfix expressions.
- Implementations often involve arrays or linked lists; space complexity considerations are key.

Read more about Commonly Asked Data Structure Interview Questions on Stack

9. Queue

- Follows the First In, First Out (FIFO) principle, ideal for task scheduling, BFS traversal, and managing buffers.
- Interview topics often involve variations like circular queue implementation, priority queues, and deque implementations.
- Optimizing queue operations is crucial in algorithms like breadth-first search.

Read more about Most Commonly Asked Data Structure Interview Questions on Queue

10. Recursion

- A method where a function calls itself to break down problems into smaller subproblems, widely used in divide and conquer algorithms.
- Interview problems often include factorial, Fibonacci, and tree/graph traversals, requiring attention to base cases and recursion depth.

Read more about Commonly Asked Data Structure Interview Questions on Recursion

11. Backtracking

- A problem-solving technique used to find all possible solutions to a problem by exploring all potential candidates and rejecting the whole recursion subtree of the solution is not feasible from a point.
- Common problems include solving Sudoku, generating permutations/combinations, and the N-Queens problem.
- Optimization involves pruning the search space and ensuring efficient exploration of possibilities.

Read more about Commonly Asked Data Structure Interview Questions on Backtracking

12. Tree

- Hierarchical structure with nodes and edges, used in searching, sorting, and representing hierarchical data like file systems.
- Interview questions often cover traversal techniques (in-order, pre-order, post-order), height balancing, and binary search trees.
- Advanced topics include AVL trees, segment trees, and tree-based dynamic programming.

Read more about Commonly Asked Interview Questions on Tree

13. Heap

- A complete binary tree used to implement priority queues, supporting efficient insertion and removal of the highest (or lowest) element.
- Interview questions often focus on implementing heaps and

- Advanced topics include heap sort and maintaining heaps during dynamic updates, such as in streaming applications.

Read more about Commonly Asked Data Structure Interview Questions on Heap Data Structure

14. Graph

- A collection of nodes (vertices) and edges, useful for modeling relationships in problems like social networks or pathfinding.
- Key problems include graph traversal (BFS, DFS), shortest path (Dijkstra, Bellman-Ford), and cycle detection.
- Advanced topics cover graph algorithms like topological sorting and minimum spanning tree (Prim's, Kruskal's).

Read more about Commonly Asked Data Structure Interview Questions on Graph

15. Dynamic Programming

- A technique for solving problems by breaking them down into simpler subproblems, storing the results to avoid redundant work.
- Interview problems often involve Fibonacci sequences, knapsack problems, and longest common subsequences.
- Optimizing space and time complexity in DP solutions is a key area of focus, particularly through memoization and tabulation.

Read more about Commonly Asked Data Structure Interview Questions on Dynamic Programming

16. Bit Manipulation

- Techniques used to manipulate individual bits of numbers, essential

- Interview questions often include problems like finding the single non-repeated element, counting set bits, and checking power of two.
- Understanding bitwise operators (AND, OR, XOR) is critical for optimizing memory usage and performance.

Read more about Commonly Asked Data Structure Interview Questions on Bit Manipulation

17. Tries

- A Trie is a tree-like data structure used for efficient retrieval of keys in a dataset of strings, often used in applications like autocomplete and spell checking.
- Common operations include insertion, search, and deletion of strings, with a time complexity of O(k) where k is the length of the string being processed.
- Tries are highly useful for solving problems involving prefix matching, longest prefix search, and dictionary-based problems, and can be optimized with techniques like compressed tries or ternary search trees.

Read more about Commonly Asked Data Structure Interview Questions on Tries

For a quick and focused revision, check out our GfG-160
Master Complete DSA from basic to advanced with DSA-360°

Related posts:

- Last Minute Notes – DS
- Commonly Asked Data Structure Interview Questions
- Commonly Asked Algorithm Interview Questions