OPERATING SYSTEM

Basics

1. What are the primary functions of an operating system?
Answer: Process management, memory management, file system management, device management, and user interface.

2. What is a kernel in an operating system?
Answer: The kernel is the core part of an OS that manages system resources, hardware, and software interactions.

3. What is the difference between monolithic and microkernel architectures?
Answer: Monolithic: All OS services run in the kernel space. Microkernel: Only essential services run in kernel space; others run in user space.

4. What is a system call?
Answer: A mechanism for user processes to request services from the OS kernel.

5. What are the differences between multitasking and multiprocessing?
Answer: Multitasking: Multiple tasks share CPU time. Multiprocessing: Multiple CPUs process tasks simultaneously.

6. What is the difference between a 32-bit and a 64-bit operating system?
Answer: A 64-bit OS can handle more RAM and process data in larger chunks compared to a 32-bit OS.

7. What is the difference between a command-line interface (CLI) and a graphical user interface (GUI)?
Answer: CLI: Text-based interface for commands. GUI: Visual interface with windows, icons, and menus.

8. What is the purpose of an interrupt in an OS?
Answer: To signal the processor about immediate attention required by hardware or software.

9. What is the difference between a real-time operating system and a general-purpose operating system?
Answer: RTOS: Ensures task completion within strict time limits. General-purpose: No strict timing constraints.

10. What is a bootloader?
Answer: A program that initializes the OS during startup.

Process Management

1. What is a process?
Answer: A process is an instance of a program in execution.

2. What are the different states of a process?

Answer: New, Ready, Running, Waiting, and Terminated.

3. What is the difference between a process and a thread?
Answer: A process is an independent execution unit with its own memory. A thread is a lightweight process that shares memory with its parent process.

4. What is context switching?
Answer: The process of saving and restoring the CPU state during a process switch.

5. Explain the difference between preemptive and non-preemptive scheduling.
Answer: Preemptive: OS can interrupt and switch processes. Non-preemptive: Processes run until completion or voluntarily yield.

6. Describe the Round Robin scheduling algorithm.
Answer: Each process is assigned a fixed time slice in a circular queue.

7. What is a zombie process?
Answer: A process that has completed execution but still has an entry in the process table.

8. What is inter-process communication (IPC)?
Answer: Mechanisms like pipes, message queues, and shared memory allow processes to communicate.

9. What are semaphores, and how are they used?
Answer: A semaphore is a synchronization tool to control access to shared resources.

10. What is a deadlock?
Answer: A situation where processes wait indefinitely for resources held by each other.

Memory Management

1. What is virtual memory?
Answer: A memory management technique that uses disk space to simulate additional RAM.

2. What is paging?
Answer: Dividing memory into fixed-size pages to manage processes efficiently.

3. What is segmentation?
Answer: Dividing memory into variable-sized segments based on logical divisions.

4. What is a page fault?
Answer: When a process tries to access a page not currently in memory.

5. What is internal and external fragmentation?
Answer: Internal: Wasted space within allocated blocks. External: Wasted space between allocated blocks.

6. What is a memory leak?

Answer: When a program fails to release allocated memory.

7. What is the role of the MMU (Memory Management Unit)?
Answer: It translates virtual addresses to physical addresses.

8. What is swapping?
Answer: Moving processes between main memory and disk.

9. What is contiguous and non-contiguous memory allocation?
Answer: Contiguous: Consecutive blocks. Non-contiguous: Scattered blocks.

10. What is demand paging?
Answer: Pages are loaded into memory only when accessed.

File Systems

1. What is the role of a file system?
Answer: To manage how data is stored and retrieved on storage devices.

2. What is an inode?
Answer: A data structure that stores metadata about a file.

3. What is the difference between FAT32 and NTFS?
Answer: FAT32: Older, limited features, lacks security. NTFS: Supports larger files, better security, and journaling.

4. What is a file descriptor?
Answer: An integer handle used by a process to access a file.

5. What are file permissions?
Answer: Rules defining who can read, write, or execute a file.

6. What is a hard link?
Answer: A directory entry that points to the same inode as the original file.

7. What is a symbolic (soft) link?
Answer: A pointer to another file or directory.

8. What is journaling in file systems?
Answer: A feature to keep track of changes for recovery in case of a crash.

9. What is a directory?
Answer: A special file that contains a list of other files and directories.

10. What is the purpose of file allocation methods?
Answer: To determine how file data is stored on disk.

11. What are the differences between contiguous, linked, and indexed allocation?

Answer: Contiguous: Consecutive blocks. Linked: Blocks linked via pointers. Indexed: Index holds pointers to all blocks.

12. What is a mount point?
Answer: A directory where a file system is attached for access.

13. What is the role of a superblock?
Answer: Stores metadata about a file system.

14. What is a file system hierarchy?
Answer: An organizational structure of files and directories.

15. What is disk partitioning?
Answer: Dividing a disk into logical sections for different file systems.

OOPS CONCEPT

Inheritance

1. **What is inheritance?**
   It allows one class (child) to inherit properties and methods from another class (parent).
2. **Types of inheritance?**
   - Single: One child inherits one parent.
   - Multiple: One child inherits from multiple parents.
   - Multilevel: A child inherits a parent who is also a child of another class.
   - Hierarchical: Multiple children inherit from one parent.
3. **Difference between inheritance and composition?**
   Inheritance uses "is-a" relationships; composition uses "has-a" relationships.
4. **What is a superclass?**
   A parent class from which other classes inherit.
5. **What is a subclass?**
   A child class that inherits from a parent class.
6. **What is the diamond problem?**
   Ambiguity arises when a class inherits from two classes that share a common parent.
7. **How to call a parent class method?**
   Use super.methodName() in Java or base.methodName() in C++.
8. **Limitations of inheritance?**
   Can cause tight coupling and increases complexity.
9. **What is the super keyword?**
   It is used to call parent class methods or constructors.
10. **Can private methods be overridden?**
    No, private methods are not inherited.

Abstraction

1. **What is abstraction?**
   Hiding complex implementation details and showing only essential features.
2. **How is it implemented?**
   Through abstract classes and interfaces.
3. **Difference between abstract classes and interfaces?**
   Abstract classes can have both abstract and concrete methods; interfaces only have abstract methods (until Java 8).
4. **Can abstract classes have constructors?**
   Yes, for initializing fields in derived classes.
5. **Difference between abstraction and encapsulation?**
   Abstraction hides implementation; encapsulation hides internal state.
6. **Example of abstraction?**
   Using an API without knowing its internal code.
7. **Why can't abstract classes be instantiated?**
   They are incomplete and meant to be extended.
8. **Role of abstract methods?**
   Enforce implementation in derived classes.
9. **Can a class be abstract and final?**
   No, because abstract classes must be extended, and final classes cannot be extended.
10. **Banking example?**
    An abstract class Account can define a method calculateInterest() while subclasses like Savings and Current implement it differently.

Polymorphism

1. **What is polymorphism?**
   Ability of an object to take many forms (e.g., method overloading, overriding).
2. **Compile-time vs. runtime polymorphism?**
   Compile-time: Method overloading.
   Runtime: Method overriding.
3. **What is method overloading?**
   Defining multiple methods with the same name but different parameters.
4. **What is method overriding?**
   A child class redefines a parent class method.
5. **Can the main method be overloaded?**
   Yes, but JVM only calls the standard version.
6. **Role of virtual keyword?**
   Enables runtime polymorphism in C++.
7. **What is dynamic method dispatch?**
   Resolving a method call at runtime based on the object's type.
8. **Real-world application of polymorphism?**
   A draw() method behaves differently for Circle and Square objects.
9. **Covariance and contravariance?**
   Refers to return types and parameter types in inheritance hierarchies.
10. **Difference between polymorphism and inheritance?**
    Polymorphism is about behavior; inheritance is about structure.

Encapsulation

1. **What is encapsulation?**
   Bundling data and methods into one unit (class) and restricting access.
2. **How is it implemented?**
   Using access modifiers like private, protected, and public.
3. **Difference between encapsulation and data hiding?**
   Data hiding is a subset of encapsulation focused on restricting data access.
4. **Role of getter and setter methods?**
   Control access to private fields.
5. **Can encapsulation work without classes?**
   No, classes are necessary for encapsulation in OOP.
6. **How does it promote loose coupling?**
   By exposing only necessary parts of a class, reducing dependencies.
7. **Access specifiers in encapsulation?**
   - private: Accessible only within the class.
   - protected: Accessible within the package and by subclasses.
   - public: Accessible everywhere.
8. **How does encapsulation enhance flexibility?**
   By allowing internal changes without affecting external code.
9. **User credentials example?**
   Use private fields for username and password, accessible via secure getter methods.
10. **Impact of breaking encapsulation?**
    Leads to unintended side effects and harder-to-maintain code.

Here are **10 interview questions and answers for each data structure topic**, useful for preparation. Each answer provides a concise solution or explanation.

Data Structures

Arrays

1. **What are the advantages and disadvantages of arrays?**
   - Advantages: Random access of elements; efficient for searching.
   - Disadvantages: Fixed size, static structure. (Source: Career Guru99)
2. **How do you find the maximum product subarray?**
   - Use two variables to track the maximum and minimum products. Update them iteratively based on current element. Complexity: O(n). (Source: Interview Prep)
3. **What is the sliding window technique?**
   - A way to find a fixed-length subarray with specific properties, reducing complexity to O(n). Example: Maximum sum of a subarray. (Source: Interview Prep)
4. **How do you merge two sorted arrays?**

- Use two pointers to iterate and merge them into a new array. Complexity: O(n). (Source: Interview Prep)

5. **What is a jagged array?**
   - An array of arrays where each subarray can have a different size. (Source: Career Guru99)
6. **How do you find the majority element?**
   - Use Boyer-Moore Voting Algorithm. Complexity: O(n). (Source: Interview Prep)
7. **What is the default value in an uninitialized array in Java?**
   - Zero for numeric, false for boolean, null for objects. (Source: Career Guru99)
8. **Explain ArrayIndexOutOfBoundsException.**
   - An error that occurs when accessing an index outside the array's bounds. (Source: Career Guru99)
9. **Can you declare an array without a size?**
   - No. Size must be defined during declaration. (Source: Career Guru99)
10. **How do you copy an array in Java?**
    - Use System.arraycopy(), Arrays.copyOf(), or a loop. (Source: Career Guru99)

Linked Lists

1. **What are the types of linked lists?**
   - Singly, doubly, and circular linked lists.
2. **How do you reverse a linked list?**
   - Use three pointers (prev, current, next) to iteratively reverse the list.
3. **What is the time complexity of searching in a linked list?**
   - O(n), as traversal is required.
4. **How do you detect a cycle in a linked list?**
   - Use Floyd's Cycle-Finding Algorithm (two-pointer technique).
5. **What are the advantages over arrays?**
   - Dynamic size and efficient insertion/deletion.
6. **How do you find the middle element?**
   - Use two pointers: one moves twice as fast as the other.
7. **How do you merge two sorted linked lists?**
   - Use recursion or iteration, comparing nodes.
8. **What is the difference between a linked list and a doubly linked list?**
   - Doubly linked lists have two pointers (prev and next), while singly linked lists only have next.
9. **How do you delete a node without head reference?**
   - Copy the data from the next node and delete it.
10. **How do you check if two linked lists intersect?**
    - Find lengths and align them; then check node by node.

Stacks

1. **What is a stack?**
   - A LIFO (Last In, First Out) data structure.
2. **How do you implement a stack using arrays?**
   - Maintain a top pointer and an array.
3. **How do you implement a stack using queues?**

- Use two queues, pushing elements into one and maintaining stack order.

4. **What are the time complexities for stack operations?**
   - Push, pop, and peek: O(1).
5. **What is stack overflow?**
   - An error when pushing onto a full stack.
6. **How do you reverse a stack?**
   - Use an auxiliary stack or recursion.
7. **How do you check balanced parentheses?**
   - Use a stack to push open parentheses and match closing ones.
8. **How do you find the minimum element in O(1)?**
   - Maintain an auxiliary stack to track minimum values.
9. **How do you convert infix to postfix expressions?**
   - Use a stack for operators and output operands in order.
10. **How do you evaluate a postfix expression?**
    - Use a stack to push operands and apply operators.

Queues

1. **What is a queue?**
   - A FIFO (First In, First Out) data structure.
2. **How do you implement a queue using stacks?**
   - Use two stacks: one for enqueue and another for dequeue.
3. **What is a circular queue?**
   - A queue where the last position connects back to the first.
4. **How do you implement a priority queue?**
   - Use a heap for efficient priority-based operations.
5. **What is the difference between a queue and a deque?**
   - A deque allows insertion/deletion from both ends.
6. **How do you find the first non-repeating character in a stream?**
   - Use a queue and a hash map.
7. **What is the time complexity for enqueue and dequeue?**
   - O(1) in a simple queue, O(log n) in a priority queue.
8. **How do you implement a k-sized sliding window maximum?**
   - Use a deque to track indices of elements.
9. **What is the application of a queue?**
   - Scheduling, buffering, and breadth-first search.
10. **How do you detect a cycle in a graph using BFS?**
    - Use a queue to traverse and track visited nodes.

Trees

1. **What is a binary tree?**
   - A tree with each node having at most two children.
2. **What is the difference between binary trees and binary search trees?**
   - Binary search trees are ordered.
3. **How do you perform in-order traversal?**
   - Left, root, right.
4. **What is a balanced tree?**
   - A tree with minimal height difference between subtrees.

5. **How do you find the height of a tree?**
    - Use recursion to calculate the maximum depth.
6. **What is the time complexity for searching in a BST?**
    - O(log n) for balanced trees.
7. **How do you find the lowest common ancestor?**
    - Traverse until nodes lie in separate subtrees.
8. **What is a trie?**
    - A tree-like data structure for storing strings efficiently.
9. **How do you perform level-order traversal?**
    - Use a queue.
10. **What is a red-black tree?**
    - A self-balancing binary search tree.

Graphs

1. **What is the difference between a graph and a tree?**
    - A tree is a connected acyclic graph.
2. **What is BFS?**
    - Level-order traversal using a queue.
3. **What is DFS?**
    - Depth-first traversal using recursion or a stack.
4. **What is a directed graph?**
    - A graph where edges have direction.
5. **What is the shortest path algorithm?**
    - Dijkstra's algorithm or Bellman-Ford.
6. **What is a cycle in a graph?**
    - A path where the start and end nodes are the same.
7. **How do you detect a cycle in a directed graph?**
    - Use DFS with a recursion stack.
8. **What is a connected graph?**
    - All nodes are reachable from any node.
9. **What is the adjacency matrix?**
    - A 2D array to represent edge connectivity.
10. **What is the time complexity of graph traversal?**
    - O(V + E) for both BFS and DFS.
    -

Algorithms.

Sorting

1. **Explain the difference between stable and unstable sorting algorithms.**
    - Stable: Maintains relative order of equal elements (e.g., Merge Sort, Bubble Sort).
    - Unstable: Doesn't guarantee order (e.g., Quick Sort, Heap Sort).

2. **What is the best sorting algorithm in terms of time complexity?**
    - For general cases: Merge Sort or Quick Sort (O(n log n)).
    - For nearly sorted arrays: Insertion Sort (O(n)).
3. **How does Quick Sort work?**
    - Uses divide-and-conquer by choosing a pivot and partitioning the array into elements less/greater than the pivot.
4. **What is the worst-case time complexity of Quick Sort?**
    - $O(n^2)$, occurs when the pivot is the smallest or largest element.
5. **When is Bubble Sort preferred?**
    - For small datasets or nearly sorted arrays (simple implementation).
6. **How is Merge Sort implemented?**
    - Recursively divides the array into halves, sorts them, and merges them back.
7. **What is the advantage of Heap Sort?**
    - Space-efficient (in-place) with O(n log n) complexity, but not stable.
8. **What is Counting Sort?**
    - A non-comparison-based sorting algorithm with O(n + k) complexity for integers.
9. **How does Radix Sort work?**
    - Processes each digit from the least to most significant using Counting Sort.
10. **What are the trade-offs of Insertion Sort?**
    - Simple and efficient for small datasets, but $O(n^2)$ for larger arrays.

Searching

1. **Explain Binary Search.**
    - Divides the sorted array and eliminates half each step; complexity: O(log n).
2. **What is Linear Search?**
    - Checks elements one by one; O(n) complexity.
3. **What is the difference between Binary and Ternary Search?**
    - Ternary divides the array into three parts instead of two, but with similar complexity.
4. **How do you search in a rotated sorted array?**
    - Use modified Binary Search to find the rotation point.
5. **What is an Interpolation Search?**

- An improved version of Binary Search that works well for uniformly distributed data; O(log log n).

6. **How do you implement a search in a BST?**
    - Recursively traverse left/right subtrees based on the key; O(h), where h is tree height.
7. **What is the complexity of searching in a hash table?**
    - Average case: O(1); Worst case: O(n) due to collisions.
8. **How do you search in a graph?**
    - Use BFS for level order or DFS for depth traversal.
9. **What is a jump search?**
    - Skips ahead by fixed steps, then performs a linear search in the smaller range; O(√n).
10. **How do you find the first occurrence of an element in a sorted array?**
    - Use modified Binary Search to focus on the first half.

Basic Complexity Analysis

1. **What is time complexity?**
    - Measures the execution time of an algorithm based on input size.
2. **What is space complexity?**
    - Measures memory usage during execution, including input, variables, and call stack.
3. **Explain Big-O Notation.**
    - Describes the upper bound of algorithm performance (e.g., O(n), O(log n)).
4. **What is the difference between $O(n^2)$ and O(n log n)?**
    - $O(n^2)$ grows faster, less efficient for large inputs compared to O(n log n).
5. **What is the complexity of Merge Sort?**
    - Time: O(n log n); Space: O(n).
6. **What is the complexity of Dijkstra's algorithm?**
    - Using a priority queue: O((V + E) log V).
7. **What are amortized complexities?**
    - Average performance over a sequence of operations (e.g., O(1) for dynamic array insertion).

8. **How do you analyze recursive algorithms?**
    - Use recurrence relations (e.g., $T(n) = 2T(n/2) + O(n)$).
9. **What is the difference between worst-case and average-case complexity?**
    - Worst-case considers maximum operations; average-case averages over all inputs.
10. **How do you calculate complexity for nested loops?**
    - Multiply the complexities of the loops (e.g., $O(n^2)$ for two nested loops of size n).

Flow Control Questions

1. **What is flow control in programming?**
    - It refers to the order in which individual statements, instructions, or function calls are executed or evaluated. Examples include sequential execution, decision-making (if, switch), and iteration (for, while).
2. **How do if and switch statements differ?**
    - if evaluates boolean expressions, while switch compares a variable to constant values. Use switch for better readability with multiple conditions on a single variable.
3. **What is the default case in a switch?**
    - It executes when no other case matches. Similar to else in if-else statements.
4. **Can you use strings in a switch?**

```
switch(day) {
  case "Monday": System.out.println("Start"); break;
  default: System.out.println("Other Day");
}
```

5. **What happens if break is omitted in a switch?**
    - The program falls through to subsequent cases.
6. **Explain the difference between break and continue.**
    - break exits a loop entirely; continue skips to the next iteration.
7. **How can you avoid deeply nested conditionals?**
    - Use guard clauses or logical operators to simplify logic.
8. **What are ternary operators?**
    - A shorthand for if-else: condition ? true_case : false_case.
9. **What is short-circuit evaluation?**
    - Logical operators like && and || evaluate left to right and stop once the result is determined.
10. **When would you use if vs. a series of if-else?**
    - Use standalone if for unrelated conditions and if-else for mutually exclusive logic.

Loop Questions

1. **What is the purpose of loops?**
    - To repeatedly execute code until a condition is met.
2. **Difference between for and while loops?**
    - for is used when iterations are known; while is for indefinite conditions.
3. **What is a do-while loop?**
    - Executes the block at least once before checking the condition.
4. **How to prevent infinite loops?**
    - Ensure the loop condition will eventually become false.
5. **How do break and continue affect loops?**
    - break exits the loop; continue skips the current iteration.
6. **How to loop through a collection safely?**
    - Use iterators or enhanced for loops to avoid modification issues.
7. **What is nested looping?**
    - Placing one loop inside another. Be cautious as it increases complexity.
8. **How can you optimize a loop for performance?**
    - Avoid unnecessary iterations, use efficient data structures, and minimize operations inside the loop.
9. **What is the difference between pre-increment and post-increment in loops?**
    - Pre-increment (++i) modifies before evaluation; post-increment (i++) evaluates before modification.
10. **How do you handle retries in loops?**
    - Use a loop with a retry counter and break on success.

Conditional Statements Questions

1. **What is the role of logical operators in conditions?**
    - Combine multiple conditions using AND (&&), OR (||), and NOT (!).
2. **What is a nested if statement?**
    - An if statement within another if, used for hierarchical conditions.
3. **What are potential issues with nested conditionals?**
    - Reduced readability and maintainability; consider refactoring.
4. **What is operator precedence in conditionals?**
    - Determines the order of evaluation. Use parentheses for clarity.

5. **How would you validate user input with conditionals?**
    - Example:
    - if len(password) < 8:
    - print("Password too short")
6. **How to combine conditions efficiently?**
    - Use logical operators or switch-case structures.
7. **What happens when a return is used inside a conditional?**
    - It exits the function immediately.
8. **What is the role of else if in decision-making?**
    - Allows for multiple mutually exclusive conditions.
9. **What is the difference between equality (==) and assignment (=) operators?**
    - == checks values; = assigns values.
10. **How to avoid common pitfalls in conditional logic?**
    - Test edge cases and use debugging tools.

Functions Questions

1. **What is a function?**
    - A reusable block of code that performs a task, defined by a name, parameters, and return type.
2. **What are the types of functions?**
    - Built-in and user-defined.
3. **What is the purpose of function arguments?**
    - To pass data into a function for processing.
4. **What is recursion?**
    - A function calling itself. Example: factorial calculation.
5. **How do you handle multiple return values?**
    - Use tuples, lists, or objects to return multiple values.
6. **What are default parameters in functions?**
    - Parameters with default values if not provided by the caller.
7. **What is the difference between pass-by-value and pass-by-reference?**
    - Pass-by-value copies data; pass-by-reference passes the address.

8. **How are global variables accessed in functions?**
    - Use the global keyword or qualifiers like this.
9. **What is function overloading?**
    - Defining multiple functions with the same name but different parameters.
10. **Why is function modularization important?**
    - Improves code readability, maintainability, and reusability.

SQL AND QUERIES

1. Basic SQL Queries

- **Q1:** How would you retrieve all records from a table?
    - **Answer:** Use the SELECT statement:
      SELECT * FROM table_name;
- **Q2:** How do you update a record in SQL?
    - **Answer:** Use the UPDATE statement:
      UPDATE table_name SET column_name = value WHERE condition;
- **Q3:** What is the purpose of the WHERE clause?
    - **Answer:** The WHERE clause filters records that meet a specific condition.
- **Q4:** How do you delete a record from a table?
    - **Answer:** Use the DELETE statement:
      DELETE FROM table_name WHERE condition;
- **Q5:** What does the COUNT() function do?
    - **Answer:** COUNT() returns the number of rows that match the given criteria.

2. Joins

- **Q1:** What is an INNER JOIN in SQL?
    - **Answer:** An INNER JOIN returns records that have matching values in both tables.

- **Q2:** How does a LEFT JOIN differ from an INNER JOIN?
    - **Answer:** A LEFT JOIN returns all records from the left table, and the matching records from the right table, while an INNER JOIN returns only matching records from both tables.
- **Q3:** What is a RIGHT JOIN?
    - **Answer:** A RIGHT JOIN is the opposite of a LEFT JOIN—it returns all records from the right table, and the matching records from the left table.
- **Q4:** What is a FULL JOIN?
    - **Answer:** A FULL JOIN returns all records when there is a match in either left or right table.
- **Q5:** How do you join three tables?
    - **Answer:** Use multiple JOIN clauses:
      SELECT * FROM table1 INNER JOIN table2 ON table1.id = table2.id INNER JOIN table3 ON table2.id = table3.id;

3. Normalization

- **Q1:** What is normalization?
    - **Answer:** Normalization is the process of organizing the attributes and relations of a database to reduce redundancy and dependency.
- **Q2:** What is the First Normal Form (1NF)?
    - **Answer:** 1NF ensures that there are no duplicate rows in a table and that each column contains atomic values.
- **Q3:** What is the Second Normal Form (2NF)?
    - **Answer:** 2NF requires that a table is in 1NF and all non-key columns are fully dependent on the primary key.
- **Q4:** What is the Third Normal Form (3NF)?
    - **Answer:** 3NF ensures that all columns are only dependent on the primary key and no transitive dependencies exist.
- **Q5:** What is Denormalization?
    - **Answer:** Denormalization involves introducing redundancy to a table by combining tables, often to improve performance.

4. Transactions

- **Q1:** What is a transaction in SQL?
    - **Answer:** A transaction is a sequence of SQL operations performed as a single unit, ensuring data integrity and consistency.
- **Q2:** What does COMMIT do in SQL?

- **Answer:** The COMMIT statement saves all changes made during the current transaction.

- **Q3:** What is a ROLLBACK?
  - **Answer:** The ROLLBACK statement undoes changes made during the current transaction.

- **Q4:** What are ACID properties?
  - **Answer:** ACID stands for Atomicity, Consistency, Isolation, and Durability, which ensure reliable transaction processing.

- **Q5:** What is a SAVEPOINT?
  - **Answer:** A SAVEPOINT allows you to set a point within a transaction to which you can later roll back.

5. Indexes

- **Q1:** What is an index in SQL?
  - **Answer:** An index improves the speed of data retrieval operations on a database table.

- **Q2:** What types of indexes are there in SQL?
  - **Answer:** There are three types of indexes: unique, clustered, and non-clustered.

- **Q3:** What is a clustered index?
  - **Answer:** A clustered index sorts the data rows in the table based on the key column. Each table can have only one clustered index.

- **Q4:** What is a non-clustered index?
  - **Answer:** A non-clustered index creates a separate structure from the data table and contains pointers to the actual data.

- **Q5:** How can you create an index?
  - **Answer:** Use the CREATE INDEX statement:
    CREATE INDEX index_name ON table_name (column_name);