**A1:**

**🔹 Q1: What is DFS (Depth-First Search)?**

**A:**  
DFS is a graph traversal algorithm that explores as far as possible along each branch before backtracking. It uses recursion or a stack to keep track of nodes.

**🔹 Q2: How is DFS implemented in your code?**

**A:**  
In my code, DFS is implemented recursively:

* It starts from a given node.
* Visits that node, marks it visited.
* Recursively visits all its unvisited neighbors.
* A Set tracks visited nodes and a List stores traversal order.

**🔹 Q3: What is the time complexity of DFS?**

**A:**  
**O(V + E)**

* Each vertex and edge is visited once.
* V = number of vertices, E = number of edges.

**🔹 Q4: What is the space complexity of DFS?**

**A:**  
**O(V + E)**

* Adjacency list: O(V + E)
* Visited set: O(V)
* Call stack: O(V) in worst case
* Traversal list: O(V)

**🔹 Q5: What is BFS (Breadth-First Search)?**

**A:**  
BFS is a graph traversal algorithm that explores all neighbors at the current level before moving to the next level. It uses a queue to manage traversal order.

**🔹 Q6: How is BFS implemented in your code?**

**A:**  
BFS is implemented recursively (which is unusual):

* Uses a queue to maintain nodes at each level.
* Visits all nodes in the queue, level by level.
* For each node, adds its unvisited neighbors to the queue.
* Tracks visited nodes and prints level-wise output.

**🔹 Q7: What is the time complexity of BFS?**

**A:**  
**O(V + E)**

* Each node is enqueued and dequeued once.
* Each edge is checked once.

**🔹 Q8: What is the space complexity of BFS?**

**A:**  
**O(V + E)**

* Adjacency list: O(V + E)
* Visited set: O(V)
* Queue: O(V) in worst case
* Traversal list: O(V)

**🔹 Q9: What is the difference between DFS and BFS?**

| **Aspect** | **DFS** | **BFS** |
| --- | --- | --- |
| Strategy | Go deep before backtracking | Explore all neighbors first |
| Data Structure | Stack (or recursion) | Queue |
| Use Case | Path finding, cycle detection | Shortest path in unweighted graph |
| Space | Less in narrow trees | More in wide graphs |

**🔹 Q10: Why did you choose to use recursive BFS?**

**A:**  
To demonstrate an alternative approach. BFS is typically iterative, but recursive BFS is implemented here for educational purposes, showing level-wise processing.

**🔹 Q11: Can DFS or BFS be used on a disconnected graph?**

**A:**  
Yes, but only one connected component will be traversed. To cover all nodes, we need to run DFS/BFS for each unvisited node separately.

**🔁 Summary Table**

| **Use Case** | **DFS or BFS?** | **Why?** |
| --- | --- | --- |
| Maze Solving | DFS | Explore all paths deeply |
| Shortest Path in Unweighted Graph | BFS | Finds shortest path in terms of hops |
| Social Networks (friend suggestions) | BFS | Level-wise connection discovery |
| Cycle Detection | DFS | Deep exploration helps detect back edges |
| Task Scheduling | DFS | For topological sorting |
| Web Crawling (wide first) | BFS | Crawls closest links first |

A2:

**✅ *1. What is the A algorithm?*\***

**Answer:**  
The A\* algorithm is a search algorithm used to find the shortest path between a start and a goal node. It combines the advantages of Dijkstra’s Algorithm and Greedy Best-First Search using a cost function:  
**f(n) = g(n) + h(n)**,  
where:

* **g(n)** is the cost from the start node to the current node.
* **h(n)** is the estimated cost from the current node to the goal node (heuristic).

**✅ 2. What heuristic is used in this code?**

**Answer:**  
The heuristic used is the **misplaced tiles** count — it counts how many tiles are not in their goal position (excluding the blank tile '0').

**✅ 3. Why is the priority queue used?**

**Answer:**  
The priority queue is used to always expand the node with the **lowest f(n) = g + h** value first. This ensures that the algorithm explores the most promising path toward the goal.

**✅ 4. What does the newNode function do?**

**Answer:**  
It creates a new node by copying the matrix and swapping the blank tile (0) to generate a new state after a valid move. It sets the cost g, and the parent node for path tracing.

**✅ 5. How do you check if the 8-puzzle is solvable?**

**Answer:**  
Using **inversion count**. The number of inversions is calculated after converting the start and goal matrices to a 1D relative format.  
If the number of inversions is even, the puzzle is solvable.

**✅ 6. What is an inversion in this context?**

**Answer:**  
An inversion is a pair of tiles (a, b) such that a appears before b in the list but a > b, based on the target positions of the tiles. It helps determine the solvability of the puzzle.

**✅ 7. What data structure represents each puzzle state?**

**Answer:**  
A class called Node is used, which holds:

* The current matrix (int[][] mat)
* The g and h costs
* Coordinates of the blank tile (x, y)
* A reference to its parent node to reconstruct the path

**✅ *8. What is the time complexity of the A algorithm?*\***

**Answer:**  
In the worst case, the time complexity is **O(b^d)**, where:

* **b** is the branching factor (number of successors per state),
* **d** is the depth of the goal node.

However, with a good heuristic, the performance improves significantly.

**✅ 9. What does printPath() do?**

**Answer:**  
It recursively prints the path from the initial state to the goal state by following the parent pointers. This shows the sequence of moves that solved the puzzle.

**✅ 10. Why are directions defined as dRow and dCol arrays?**

**Answer:**  
To simplify moving the blank tile (0) in 4 directions: **up, down, left, and right**. Each direction is represented by corresponding changes in row and column indexes.

**✅ 1. What is the difference between informed and uninformed search strategies?**

**Answer:**  
Informed search strategies use **problem-specific knowledge**, such as a heuristic function, to find solutions more efficiently. Uninformed search strategies (like BFS or DFS) do not use such knowledge and blindly explore the search space.

**✅ *2. Why is A called an informed or heuristic search algorithm?*\***

**Answer:**  
Because it uses a heuristic function h(n) in combination with the actual cost g(n) to estimate the total cost f(n) = g(n) + h(n) for reaching the goal. This makes the search more directed and efficient.

**✅ *3. What is the role of the heuristic function in A algorithm?*\***

**Answer:**  
The heuristic function h(n) estimates the cost from the current node to the goal. It helps the algorithm prioritize nodes that appear to be closer to the goal.

**✅ 4. What is an admissible heuristic?**

**Answer:**  
A heuristic is **admissible** if it **never overestimates** the actual cost to reach the goal. This ensures that the A\* algorithm finds the optimal solution.

**✅ 5. What does the function f(n) = g(n) + h(n) represent?**

**Answer:**  
It represents the **estimated total cost** of a path through node n to the goal:

* g(n) is the actual cost to reach node n from the start,
* h(n) is the estimated cost from n to the goal.

**✅ 6. Can you explain g(n) and h(n) with an example in a game-like context?**

**Answer:**  
Yes. In a tile-based game:

* g(n) could be the number of moves taken from the start to the current square.
* h(n) could be the Manhattan distance (estimated steps) from the current square to the goal square.

**✅ 7. What happens if the heuristic function overestimates the cost to the goal?**

**Answer:**  
If the heuristic overestimates, it is **not admissible**, and A\* may **not find the optimal solution**, making the algorithm less reliable.

**✅ *8. What is meant by "path score" in A?*\***

**Answer:**  
Path score refers to the value of f(n) = g(n) + h(n) assigned to each node. It determines which node will be expanded next — the one with the lowest path score is chosen.

**✅ *9. Why is heuristic design important in A algorithm?*\***

**Answer:**  
Because the **efficiency and correctness** of the A\* algorithm heavily depend on how accurate and admissible the heuristic function is.

**✅ 10. What is meant by saying a heuristic function provides “domain knowledge”?**

**Answer:**  
It means the heuristic incorporates knowledge **specific to the problem domain** (e.g., how tiles behave in an 8-puzzle), which helps guide the search process more intelligently.

**✅ *11. Is A guaranteed to find the optimal solution?*\***

**Answer:**  
Yes, **if the heuristic function is admissible and consistent**, A\* is guaranteed to find the shortest (optimal) path.

**✅ 12. What are examples of heuristic functions for the 8-puzzle?**

**Answer:**

* **Misplaced Tiles Heuristic**: counts the number of tiles not in the correct position.
* **Manhattan Distance Heuristic**: sums the distances of tiles from their goal positions

A3:

**🔹 Basic Conceptual Questions**

**Q1: What is a Minimum Spanning Tree (MST)?**  
**A:** A Minimum Spanning Tree is a subset of edges from a connected, weighted, undirected graph that connects all vertices without forming cycles and has the minimum possible total edge weight.

**Q2: What is the difference between Prim’s and Kruskal’s algorithm?**  
**A:**

* **Prim’s Algorithm** starts from a vertex and grows the MST by adding the minimum weight edge connecting the MST to a new vertex.
* **Kruskal’s Algorithm** sorts all edges by weight and adds the shortest edge to the MST that doesn’t form a cycle, using union-find to check cycles.

**Q3: Which data structures are used in Prim’s algorithm here?**  
**A:**

* An array key[] to store minimum edge weight.
* A boolean array mstSet[] to track included vertices.
* A parent[] array to store the MST structure.

**Q4: Which data structures are used in Kruskal’s algorithm here?**  
**A:**

* A list of Edge objects to store all edges.
* Arrays parent[] and rank[] for union-find operations with path compression and union by rank.

**Q5: How do you detect a cycle in Kruskal’s algorithm?**  
**A:**  
Using the **union-find** data structure. If two vertices of an edge belong to the same set (i.e., have the same parent), adding that edge would form a cycle.

**🔹 Code-Based Questions**

**Q6: Why do we initialize key[0] = 0 in Prim’s algorithm?**  
**A:**  
It ensures that the first vertex is picked first since it's the starting point of the MST.

**Q7: What is the purpose of the minKey() function?**  
**A:**  
It returns the index of the vertex with the smallest key[] value not yet included in MST, to decide which vertex to add next.

**Q8: Why do we sort edges before applying Kruskal’s algorithm?**  
**A:**  
Sorting ensures we always consider the smallest weight edge first, which is key to forming a minimum spanning tree.

**Q9: What is the time complexity of Prim’s and Kruskal’s algorithms in your implementation?**  
**A:**

* **Prim’s:** O(V2)O(V^2)O(V2) since we use an adjacency matrix and linear search in minKey().
* **Kruskal’s:** O(Elog⁡E+E⋅α(V))O(E \log E + E \cdot \alpha(V))O(ElogE+E⋅α(V)), where α(V)\alpha(V)α(V) is the inverse Ackermann function (due to union-find).

**Q10: Can Prim's and Kruskal's algorithms handle negative weights?**  
**A:**  
Yes, both algorithms work with negative weights as long as the graph has no directed cycles and is undirected and connected.

**🔹 Real-World Application Questions**

**Q11: Where is MST used in real-world scenarios?**  
**A:**

* Designing network cables to minimize cost
* Road and railway network planning
* Clustering in machine learning
* Image segmentation

**Q12: What happens if the input graph is disconnected?**  
**A:**  
MST cannot be formed since there’s no way to connect all vertices. The result would be a Minimum Spanning **Forest**, not a Tree.

A4:

**ChatGPT said:**

Here are some **viva questions and answers** based on your N-Queens Java code using both **Backtracking** and **Branch and Bound** techniques:

**🔸 Basic Understanding**

**Q1. What is the N-Queens problem?**  
**A:** The N-Queens problem involves placing N queens on an N×N chessboard such that no two queens threaten each other. This means no two queens can share the same row, column, or diagonal.

**Q2. What approach have you used to solve this problem?**  
**A:** I have implemented two approaches:

* **Backtracking**: A brute-force recursive method that places queens row by row and backtracks upon conflict.
* **Branch and Bound**: An optimized version that avoids checking for conflicts by maintaining arrays to track column and diagonal attacks.

**🔸 Code-Specific Questions**

**Q3. What does the isSafe() function check?**  
**A:** It checks whether a queen can be safely placed at a given position without being attacked by another queen on the same column or diagonals above the current row.

**Q4. Why do we not check for row conflicts in isSafe()?**  
**A:** We place one queen per row, so there's no need to check for row conflicts.

**Q5. What is backtracking in your code?**  
**A:** Backtracking refers to the process of removing a previously placed queen when no valid placement is found in the next row. This helps explore other possibilities to find a solution.

**Q6. How is Branch and Bound different from Backtracking here?**  
**A:** In Branch and Bound, we use three arrays (colConflicts, diag1Conflicts, and diag2Conflicts) to track conflicts in constant time, which avoids scanning the board like in backtracking.

**🔸 Complexity and Optimization**

**Q7. What is the time complexity of the Backtracking solution?**  
**A:** The worst-case time complexity is **O(N!)**, as we try N possibilities in each of N rows in the worst case.

**Q8. How does Branch and Bound improve performance?**  
**A:** It avoids repeated conflict checks by using auxiliary arrays to instantly know if a column or diagonal is under attack, thus reducing unnecessary recursive calls.

**🔸 Implementation and Output**

**Q9. What does your printBoard() function do?**  
**A:** It prints the board configuration, marking 'Q' where queens are placed and '.' for empty spaces.

**Q10. Why do you print the board after each placement and backtrack?**  
**A:** To visualize the step-by-step process of how queens are placed and removed during the recursive calls. It helps in debugging and understanding the flow.

**🔸 Additional Conceptual Questions**

**Q11. Can this solution be extended to find all possible solutions?**  
**A:** Yes. By modifying the base case in recursion to not return immediately after one solution is found, and by collecting all valid boards, we can list all possible solutions.

**Q12. How many solutions exist for N=4?**  
**A:** There are **2 distinct solutions** to the 4-Queens problem

**✅ Definition:**

A **Constraint Satisfaction Problem** is a problem where the goal is to **find values for variables** from given domains such that **all constraints between the variables are satisfied**.

**✅ Components of a CSP:**

1. **Variables** – X1,X2,...,XnX\_1, X\_2, ..., X\_nX1​,X2​,...,Xn​  
   (e.g., for 4-queens: positions of 4 queens)
2. **Domains** – Possible values each variable can take  
   (e.g., row numbers: {1, 2, 3, 4})
3. **Constraints** – Rules that restrict the values the variables can take together  
   (e.g., no two queens in the same row, column, or diagonal)

In the context of the N-Queens problem, the **time complexity** typically depends on the algorithm used to solve it. Let's break it down for different approaches:

**1. Backtracking Algorithm:**

* In backtracking, for each row, you try placing a queen in all available columns, checking if the placement is safe. If placing a queen in one column leads to no valid solution, you backtrack and try the next column.
* For an **N×N** board, for each of the **N** rows, you attempt to place a queen in each of the **N** columns. The number of possibilities is reduced by checking for conflicts (same column or diagonal attacks) as you go.
* **Worst-case time complexity**:  
  The worst-case scenario occurs when you try all possible placements for each row. For each row, you try **N** possibilities, leading to a worst-case time complexity of **O(N!)** because you are recursively trying all possible placements for the queens across all rows.

**2. Branch and Bound:**

* This is an optimization technique that prunes the search tree more efficiently by eliminating invalid positions early (using the constraints of the N-Queens problem such as column, diagonal conflicts).
* While it might improve performance in practice, in the **worst case**, it still doesn't change the fundamental complexity, which can still be considered **O(N!)** in the case of an exhaustive search.

**Summary:**

* **Backtracking** (without optimization) has a **time complexity of O(N!)**, because in the worst case, we explore all possible queen placements across all rows.

A5:

**1. What is the purpose of using the SentenceTransformer model in your chatbot?**

**Answer:**  
The SentenceTransformer model is used to convert user input and predefined patterns into numerical representations (embeddings). These embeddings are used to measure the semantic similarity between the input and the available patterns. By comparing the embeddings of the user input to the embeddings of the patterns, the chatbot can identify the most relevant intent and provide an appropriate response.

**2. How does the chatbot handle user input and determine the intent?**

**Answer:**  
The chatbot preprocesses the user input (removes special characters and converts it to lowercase), then encodes it into an embedding using the SentenceTransformer model. The chatbot compares this embedding with the embeddings of predefined patterns in the intent dictionary using cosine similarity. The intent with the highest similarity is chosen, and a corresponding response is generated. If the input matches the "order\_status" intent, the chatbot expects an order ID in the next message.

**3. Why is the cosine similarity used in this code?**

**Answer:**  
Cosine similarity is used to measure the similarity between two vectors (in this case, embeddings). It computes the cosine of the angle between the vectors, with values closer to 1 indicating higher similarity. By using cosine similarity, the chatbot can identify which predefined pattern is most similar to the user's input and choose the appropriate response.

**4. How does the bot handle order-related queries, such as checking the order status?**

**Answer:**  
When the user asks about their order status (matching the "order\_status" intent), the chatbot sets a flag (expecting\_order\_id) to True. This flag indicates that the bot is now expecting the user to provide an order ID. Once the user provides the ID, the bot uses it to generate a response with a link to the orders page, where the user can check the status of their order.

**5. What happens if the chatbot doesn't understand the user's query?**

**Answer:**  
If the confidence of the best match (calculated using cosine similarity) is below a certain threshold (in this case, 0.5), the chatbot responds with a message like, "Sorry, I didn't quite get that. Could you please rephrase your request?" This indicates that the chatbot was unable to identify a matching intent with high confidence.

**6. Why is the random.choice() function used when selecting responses?**

**Answer:**  
The random.choice() function is used to randomly select a response from the list of possible responses for a given intent. This helps to make the conversation feel more dynamic and natural, as the chatbot can offer a variety of different responses to the same input.

**7. What is the purpose of using tkinter in your chatbot?**

**Answer:**  
tkinter is used to create the graphical user interface (GUI) for the chatbot. It provides an easy way to build a window with a chat area where the conversation is displayed, and a text input field where the user can type their messages. The chatbot's responses are shown in the chat area, and the user can interact with it through the input field.

**8. What is the role of the context dictionary in the code?**

**Answer:**  
The context dictionary is used to track the state of the conversation. Specifically, it helps the chatbot determine whether it's expecting an order ID (after an order status query) and stores the last identified intent. This allows the chatbot to maintain some context between user messages and generate more relevant responses.

**9. What is the significance of the preprocess function in this code?**

**Answer:**  
The preprocess function cleans the user's input by converting it to lowercase and removing any non-alphanumeric characters. This ensures that the chatbot can process input in a uniform format, regardless of case or punctuation, making the intent recognition more accurate.

**10. Can the chatbot be further improved? If so, how?**

**Answer:**  
Yes, the chatbot can be improved in several ways:

* **Handling more complex conversations:** The bot could be enhanced to handle more advanced conversations, maintain a deeper context, and handle multi-turn interactions.
* **Better intent detection:** More sophisticated models or techniques, such as fine-tuning the SentenceTransformer model, could improve the bot's ability to detect a wider range of intents.
* **Database integration:** The bot could be integrated with a database to track user queries or orders in real-time, allowing it to provide more personalized responses.
* **Improved fallback mechanism:** Instead of a simple "Sorry, I didn’t quite get that" message, the bot could offer suggestions or ask clarifying questions to guide the user.

A6:

**Concepts related Theory:**

**What is an expert system?**

An expert system is a computer program that uses artificial intelligence (AI) technologies to

simulate the judgment and behavior of a human or an organization that has expertise and

experience in a particular field.

Expert systems are usually intended to complement, not replace, human experts.

**How does an expert system work?**

Modern expert knowledge systems use machine learning and artificial intelligence to

simulate the behavior or judgment of domain experts. These systems can improve their

performance over time as they gain more experience, just as humans do.

Expert systems accumulate experience and facts in a knowledge base and integrate them with

an inference or rules engine -- a set of rules for applying the knowledge base to situations

provided to the program.

The inference engine uses one of two methods for acquiring information from the knowledge

base:

1. **Forward chaining** reads and processes a set of facts to make a logical prediction about

what will happen next. An example of forward chaining would be making predictions

about the movement of the stock market.

2. **Backward chaining** reads and processes a set of facts to reach a logical conclusion about

why something happened. An example of backward chaining would be examining a set

of symptoms to reach a medical diagnosis.

An expert system relies on having a good knowledge base. Experts add information to the

knowledge base, and non-experts use the system to solve complex problems that would

usually require a human expert.

P:F-LTL-UG/03/R1

The process of building and maintaining an expert system is called knowledge engineering.

Knowledge engineers ensure that expert systems have all the necessary information to solve a

problem. They use various knowledge representation methodologies, such as symbolic

patterns, to do this. The system's capabilities can be enhanced by expanding the knowledge

base or creating new sets of rules.

**What are the components of an expert system?**

There are three main components of an expert system:

● **The knowledge base.** This is where the information the expert system draws upon is

stored. Human experts provide facts about the expert system's particular domain or

subject area are provided that are organized in the knowledge base. The knowledge base

often contains a knowledge acquisition module that enables the system to gather

knowledge from external sources and store it in the knowledge base.

● **The inference engine.** This part of the system pulls relevant information from the

knowledge base to solve a user's problem. It is a rules-based system that maps known

information from the knowledge base to a set of rules and makes decisions based on those

inputs. Inference engines often include an explanation module that shows users how the

system came to its conclusion.

● **The user interface.** This is the part of the expert system that end users interact with to

get an answer to their question or problem.

**Applications of expert systems**

These systems have played a large role in many industries, including the following:

● **Financial services,** where they make decisions about asset management, act

as robo-advisors and make predictions about the behavior of various markets and other

financial indicators.

● **Mechanical engineering,** where they troubleshoot complex electromechanical

machinery.

 **Why is Python used for chatbot development?**

* **Answer**: Python is widely used for chatbot development due to its simplicity, readability, and extensive support for AI and Natural Language Processing (NLP). It has several libraries and frameworks like NLTK (Natural Language Toolkit), spaCy, and TensorFlow, which make it easier to implement machine learning algorithms and NLP models. These tools help in processing and understanding user inputs, making Python an ideal choice for building intelligent chatbots.

 **What are the requirements for developing a chatbot?**

* **Answer**: To develop a chatbot, the following requirements are needed:
  1. **Business Goal and Use Case**: Clearly define the purpose of the chatbot, whether it's for customer service, lead generation, or another function.
  2. **Technology Stack**: Choose the programming language (like Python), libraries (such as NLTK or spaCy), and chatbot platforms (like Dialogflow, Rasa, or Wit.ai).
  3. **Data**: Data for training the chatbot (e.g., conversational data, FAQs, customer queries).
  4. **NLP Techniques**: Implement NLP for understanding the user input and generating accurate responses.
  5. **Integration**: Integrate the chatbot with platforms (websites, mobile apps, social media, etc.).
  6. **User Interface**: Develop the user interface (text-based or voice-based).
  7. **Testing and Optimization**: Ensure thorough testing and optimization for better performance.

 **How do you evaluate a chatbot’s performance?**

* **Answer**: Chatbot performance can be evaluated through several metrics:
  1. **Accuracy**: How accurately does the chatbot understand and respond to user queries?
  2. **User Satisfaction**: User feedback (ratings, reviews, and satisfaction surveys) can provide insights into how well the chatbot is performing.
  3. **Response Time**: The time taken for the chatbot to process and respond to user queries.
  4. **Engagement**: Measure user interactions, such as the number of conversations, queries resolved, and user retention rates.
  5. **Error Rate**: Analyze the number of incorrect responses or failure to understand user input.

 **How do I improve my chatbot’s accuracy?**

* **Answer**: To improve a chatbot’s accuracy, consider the following steps:
  1. **Improve Training Data**: Use more high-quality, diverse, and relevant training data to help the chatbot understand different variations of queries.
  2. **Use NLP Models**: Implement advanced NLP models (like transformers or BERT) for better understanding of context and intent.
  3. **Regularly Update the Bot**: Continuously update the bot with new information and training data to handle evolving conversations.
  4. **Implement Feedback Loops**: Collect feedback from users and improve the bot’s responses based on their input.
  5. **Monitor and Optimize**: Regularly analyze the chatbot’s performance and optimize algorithms to handle edge cases and improve accuracy.

CC:

**🌐 1. What is Salesforce?**

**Answer:** Salesforce is a cloud-based Customer Relationship Management (CRM) platform that helps businesses manage sales, customer service, marketing, and more through apps built using its tools and services.

**📦 2. What is an Object in Salesforce?**

**Answer:** An object in Salesforce is a table that stores data. There are two types: **Standard Objects** (like Account, Contact) and **Custom Objects** (created by users for specific needs).

**🧱 3. What is the difference between a Standard Object and a Custom Object?**

**Answer:**

* **Standard Object:** Provided by Salesforce by default (e.g., Account, Contact).
* **Custom Object:** Created by users to store data unique to their business process.

**🔗 4. What is the difference between Lookup and Master-Detail relationships?**

**Answer:**

* **Lookup Relationship:** Loosely coupled; child can exist without the parent.
* **Master-Detail Relationship:** Tightly coupled; child record depends on the parent and is deleted when the parent is deleted.

**🛠️ 5. What are Validation Rules in Salesforce?**

**Answer:** Validation rules ensure data integrity by preventing users from saving invalid data. They evaluate expressions and display error messages when conditions are not met.

**⚙️ 6. What is a Workflow Rule?**

**Answer:** A Workflow Rule is an automation tool that performs actions like field updates, email alerts, or task creation when certain conditions are met.

**🌀 7. What is Flow in Salesforce? How is it different from Workflow?**

**Answer:** Flow is a powerful automation tool that allows complex logic, user input, and multiple-step processes. Unlike Workflow, which is limited in actions, Flow supports loops, decisions, and record creation/update/deletion.

**🧪 8. What is a Sandbox in Salesforce?**

**Answer:** A Sandbox is a copy of your production environment used for development, testing, and training without affecting live data.

**🔐 9. What is the difference between Profile and Permission Set?**

**Answer:**

* **Profile:** Defines base-level access for users (only one profile per user).
* **Permission Set:** Grants additional permissions on top of the profile (users can have multiple permission sets).

**📊 10. What are Reports and Dashboards in Salesforce?**

**Answer:**

* **Report:** A list of records that meet specific criteria, shown in rows and columns.
* **Dashboard:** A visual representation of reports using charts, gauges, and tables.

**📂 11. What is the App Launcher in Salesforce?**

**Answer:** The App Launcher is a menu in Lightning Experience that allows users to access different apps and items like objects, tabs, and pages.

**📋 12. What is a Page Layout in Salesforce?**

**Answer:** Page Layouts control the organization of fields, buttons, and related lists on object record pages. Different layouts can be assigned to different profiles.

**🔁 13. What are Record Types in Salesforce?**

**Answer:** Record Types allow different business processes, picklist values, and page layouts to be applied to the same object based on criteria like user profile or use case.

**📥 14. What is a Change Set?**

**Answer:** A Change Set is a collection of components (objects, fields, workflows, etc.) used to deploy customizations from one Salesforce org to another (usually sandbox to production).

**💡 15. What is SOQL?**

**Answer:** SOQL (Salesforce Object Query Language) is used to retrieve data from Salesforce objects similar to SQL but specific to Salesforce’s data model.

**☁️ 1. What is Cloud Computing?**

**Answer:** Cloud Computing is the delivery of computing services like servers, storage, databases, networking, software, and analytics over the internet (“the cloud”) to offer faster innovation, flexible resources, and economies of scale.

**🧠 2. What are the main types of Cloud Computing?**

**Answer:**

* **Public Cloud:** Services are offered over the public internet and shared across multiple users (e.g., AWS, Azure).
* **Private Cloud:** Cloud infrastructure is operated solely for one organization.
* **Hybrid Cloud:** A combination of public and private cloud resources.

**🏗️ 3. What are the service models of Cloud Computing?**

**Answer:**

* **IaaS (Infrastructure as a Service):** Provides virtualized computing resources over the internet.
* **PaaS (Platform as a Service):** Provides a platform allowing customers to develop, run, and manage applications.
* **SaaS (Software as a Service):** Delivers software applications over the internet on a subscription basis.

**🧩 4. Give examples of IaaS, PaaS, and SaaS.**

**Answer:**

* **IaaS:** Amazon EC2, Microsoft Azure VM
* **PaaS:** Google App Engine, Heroku
* **SaaS:** Google Workspace (Gmail, Docs), Salesforce, Dropbox

**🔒 5. What are the advantages of Cloud Computing?**

**Answer:**

* Cost-effective
* Scalable and flexible
* Easy backup and recovery
* High availability and performance
* Accessibility from anywhere

**📉 6. What are the disadvantages of Cloud Computing?**

**Answer:**

* Data security and privacy concerns
* Internet dependency
* Limited control over infrastructure
* Potential downtime

**🛠️ 7. What is virtualization in Cloud Computing?**

**Answer:** Virtualization is the process of creating a virtual version of something, like servers or storage devices, which allows multiple operating systems to run on a single physical machine.

**📍 8. What is the difference between Scalability and Elasticity?**

**Answer:**

* **Scalability:** Ability to increase resources as demand grows (long-term).
* **Elasticity:** Ability to automatically increase or decrease resources based on current demand (short-term, dynamic).

**📦 9. What is a Virtual Machine (VM)?**

**Answer:** A VM is a software emulation of a physical computer that runs an operating system and applications just like a real computer.

**🧑‍💻 10. What is a Cloud Service Provider?**

**Answer:** A Cloud Service Provider (CSP) is a company that offers cloud services like storage, databases, servers, and networking. Examples: AWS, Microsoft Azure, Google Cloud Platform (GCP).

**☁️ 1. What is the difference between cloud computing and traditional computing?**

**Answer:**

* **Cloud computing** delivers resources over the internet on-demand, with scalability and no need for local infrastructure.
* **Traditional computing** requires on-premise hardware and software, which are managed and maintained by the user.

**🧳 2. What are the characteristics of cloud computing?**

**Answer:**

* **On-demand self-service**: Users can provision resources as needed.
* **Broad network access**: Accessible over the internet from various devices.
* **Resource pooling**: Shared resources among multiple users.
* **Rapid elasticity**: Resources can be scaled up/down quickly.
* **Measured service**: Usage is monitored and billed accordingly.

**🔒 3. How is data secured in cloud computing?**

**Answer:**  
Cloud security involves:

* **Encryption (at rest and in transit)**
* **Identity and Access Management (IAM)**
* **Firewalls and Virtual Private Clouds (VPCs)**
* **Multi-factor authentication (MFA)**
* **Regular audits and compliance standards** (e.g., ISO 27001, GDPR)

**💾 4. What is a hypervisor?**

**Answer:**  
A **hypervisor** is software that creates and manages **virtual machines (VMs)** by separating the physical hardware from the operating systems.

* **Type 1 (Bare-metal)**: Runs directly on hardware (e.g., VMware ESXi)
* **Type 2 (Hosted)**: Runs on a host OS (e.g., Oracle VirtualBox)

**🏘️ 5. What is multi-tenancy in cloud computing?**

**Answer:**  
Multi-tenancy is a model where **a single instance of software serves multiple users (tenants)**. Each tenant’s data is isolated and secure, but they share the same infrastructure.

**🚨 6. What is cloud bursting?**

**Answer:**  
Cloud bursting is a hybrid cloud setup where **an application runs in a private cloud but can burst into a public cloud** when the demand for computing capacity spikes.

**🔁 7. What is load balancing in cloud computing?**

**Answer:**  
Load balancing is the process of **distributing workloads across multiple servers** to ensure no single server is overwhelmed, improving performance and availability.

**⚙️ 8. What is an API in cloud computing?**

**Answer:**  
An API (Application Programming Interface) is a set of rules that **allows applications to interact with cloud services** (e.g., AWS SDKs, REST APIs for managing cloud storage or VMs).

**📤 9. What is auto-scaling?**

**Answer:**  
Auto-scaling is a feature in cloud platforms (like AWS Auto Scaling) that **automatically adjusts the number of compute resources** based on real-time demand.

**🧱 10. What is containerization, and how is it different from virtualization?**

**Answer:**

* **Containerization** (e.g., Docker) packages applications with their dependencies and runs them in isolated environments using the host OS kernel.
* **Virtualization** creates full-fledged VMs with separate OSes using a hypervisor.
* **Containers are lightweight**, faster to start, and use fewer resources than VMs.

**🗄️ 11. What is cloud storage? Name its types.**

**Answer:**  
Cloud storage allows users to store and access data over the internet.  
**Types include:**

* **Object Storage** (e.g., AWS S3)
* **Block Storage** (e.g., AWS EBS)
* **File Storage** (e.g., AWS EFS)

**📍 12. What is a region and availability zone in cloud platforms like AWS?**

**Answer:**

* **Region:** A geographical area (e.g., Asia Pacific - Mumbai)
* **Availability Zone (AZ):** One or more isolated data centers within a region that ensure fault tolerance and high availability.

**📑 13. What is serverless computing?**

**Answer:**  
Serverless computing allows developers to **run code without managing servers**. Cloud providers handle provisioning and scaling.  
**Example:** AWS Lambda, Azure Functions

**📊 14. How is billing calculated in cloud computing?**

**Answer:**  
Cloud billing is usually based on:

* **Compute usage (vCPU hours)**
* **Storage consumed (per GB)**
* **Data transfer/bandwidth**
* **API or function call counts (for serverless)**  
  Cloud providers offer **cost calculators** and **billing dashboards**.