AI CONCEPTS PREP DECK

20 AI Concepts You Must Understand in 2026

Detailed explanations, interview prompts, and answer hints for PMs, builders, founders, and operators.

40 concept slides

Two per concept: deep notes plus interview questions

60 answer hints

Three questions per concept with structured hints

1 answer pattern

Define. Explain. Example. Limitation. Use it every time.

HOW TO USE THIS DECK

Use it like a prep system, not a glossary.

1. Learn

Read the one-line definition first. Then use the explanation blocks to understand what the concept actually does in a product or system.

2. Speak

Practice explaining each concept out loud in 60 seconds. If you cannot say it simply, revisit the analogy and common trap.

3. Interview

Use the question slide after each concept. Give your answer first, then compare against the hints and tighten your response.

Best rhythm: 5 concepts per day, then one mock interview that mixes technical depth with product judgment.

PART 01

The Foundation

How raw input becomes model behavior.

01 Neural Networks

02 Tokenization

03 Embeddings

04 Attention

05 Transformers

Neural Networks

A neural network is a layered function that learns useful patterns by adjusting connection weights.

Detailed explanation

- A neural network takes inputs, passes them through layers, and produces an output. Each layer transforms the signal into a more useful representation.
- The weights are the learned parameters. During training, the model compares its prediction to the correct answer, computes error, and updates weights to reduce future error.
- Modern AI systems scale this simple idea to billions or trillions of parameters. The surprising part is that simple math, repeated at massive scale, can learn language, vision, code, and strategy.

Why it matters

- It is the primitive behind almost every modern model.
- It explains why data, compute, architecture, and training objective matter.
- It helps you reason about why models improve gradually, not through explicit rules.

Common trap

Do not describe neurons as literal brain cells. They are mathematical units inspired by biology, not a faithful brain simulation.

Mental model

Think of it as a huge adjustable mixing board. Training turns the knobs until the output sounds right.

Neural Networks: questions and answer hints

Q1

How would you explain a neural network to a non-technical stakeholder?

ANSWER HINTS

- Use layers, weights, training, and prediction in plain language.
- Mention that it learns patterns from examples instead of following hand-written rules.
- Avoid brain metaphors that imply human-like understanding.

Q2

What happens during training?

ANSWER HINTS

- The model predicts, measures error, and updates weights.
- Backpropagation sends error signals backward through the network.
- The goal is lower loss on future examples, not memorizing one answer.

Q3

Why does scale matter for neural networks?

ANSWER HINTS

- More parameters can represent richer patterns.
- More data and compute help train those parameters.
- Scale helps, but architecture, data quality, and evaluation still matter.

Answer pattern: define the concept, explain why it matters, give one concrete product example, then name one limitation or risk.

Tokenization

Tokenization converts text into chunks that a model can process as numbers.

Detailed explanation

- Models do not read words the way humans do. Text is split into tokens, which can be words, word fragments, punctuation, spaces, or characters.
- Each token gets mapped to an ID. The model then works with those IDs and their vector representations.
- Tokenization affects cost, context length, multilingual performance, and how models handle rare words, code, names, and formatting.

Why it matters

- It is the first step in every LLM request.
- It explains why short-looking text can still use many tokens.
- It affects prompt design, latency, context limits, and pricing.

Common trap

Do not assume one word equals one token. That is often wrong, especially for code, numbers, and non-English text.

Mental model

Think of tokenization like cutting a sentence into reusable Lego pieces before building meaning.

Tokenization: questions and answer hints

Q1

Why do LLM APIs charge by tokens instead of words?

ANSWER HINTS

- Tokens are the model's actual processing units.
- Different languages and formats tokenize differently.
- Input and output tokens both consume compute.

Q2

How can tokenization affect model behavior?

ANSWER HINTS

- Rare names and unusual strings may split into many fragments.
- Formatting and whitespace can change token patterns.
- Long prompts can hit context limits faster than expected.

Q3

What should product teams remember about tokens?

ANSWER HINTS

- Budget for prompts, retrieved context, tool outputs, and final answers.
- Optimize repeated boilerplate.
- Measure token use in real workflows, not just sample prompts.

Answer pattern: define the concept, explain why it matters, give one concrete product example, then name one limitation or risk.

Embeddings

Embeddings represent meaning as vectors so similar ideas sit near each other in space.

Detailed explanation

- An embedding converts text, images, or other data into a list of numbers. The list captures semantic features learned from training data.
- Items with similar meaning end up close together by vector distance. This makes it possible to search by meaning instead of exact keywords.
- Embeddings power semantic search, recommendations, clustering, deduplication, and retrieval for RAG systems.

Why it matters

- They are the bridge between language and computation.
- They enable search for meaning, not just string matching.
- They are foundational for vector databases and retrieval systems.

Common trap

Do not say embeddings understand meaning like a person. They encode statistical relationships that are useful for matching and comparison.

Mental model

Imagine every idea has coordinates on a map. Related ideas have nearby addresses.

Embeddings: questions and answer hints

Q1

What is an embedding and why is it useful?

ANSWER HINTS

- It is a vector representation of content.
- Similarity can be computed with distance metrics.
- It supports semantic search and retrieval.

Q2

How do embeddings differ from keyword search?

ANSWER HINTS

- Keyword search matches literal terms.
- Embedding search can match related meaning.
- Keyword search can be more precise for exact IDs, codes, or names.

Q3

What can go wrong with embeddings?

ANSWER HINTS

- Bad chunking can hide the relevant context.
- Domain-specific meaning may need better models or metadata filters.
- Similarity is not the same as factual correctness.

Answer pattern: define the concept, explain why it matters, give one concrete product example, then name one limitation or risk.

Attention

Attention lets each token weigh which other tokens matter most for the current prediction.

Detailed explanation

- Attention computes relationships between tokens. A token can attend strongly to nearby or distant tokens depending on what helps the model predict the next token.
- This is why models can connect references, preserve context, and handle long-range dependencies better than earlier sequence models.
- Self-attention means every token in a sequence can compare itself with other tokens in that same sequence.

Why it matters

- It is the core mechanism inside transformers.
- It explains why context order and wording can matter.
- It helps models connect entities, instructions, and constraints.

Common trap

Do not oversimplify attention as memory. It is a weighting mechanism, not a durable database.

Mental model

Think of attention like highlighting the parts of a document that matter for answering the next question.

Attention: questions and answer hints

Q1

What problem did attention help solve?

ANSWER HINTS

- It improved handling of long-range dependencies.
- It allowed models to focus on relevant tokens.
- It enabled more parallel training than older recurrent designs.

Q2

What is self-attention?

ANSWER HINTS

- Tokens compare with other tokens in the same input.
- The model computes relevance scores.
- Those scores shape the next internal representation.

Q3

Is attention the same as explanation?

ANSWER HINTS

- Not necessarily.
- Attention weights can be informative but are not full causal explanations.
- Use evaluation and interpretability tools for stronger claims.

Answer pattern: define the concept, explain why it matters, give one concrete product example, then name one limitation or risk.

Transformers

Transformers stack attention and feed-forward layers to model sequences at scale.

Detailed explanation

- A transformer is an architecture built around self-attention. It processes tokens in parallel and learns relationships across the sequence.
- Large language models are usually decoder-only transformers trained to predict the next token. Other transformer variants are used for encoding, translation, vision, and multimodal work.
- Transformers became dominant because they scale well with data and compute, and they handle context better than many earlier architectures.

Why it matters

- They are the architecture behind most frontier LLMs.
- They explain the connection between training objective and generation.
- They are flexible across text, code, images, audio, and video.

Common trap

Do not say transformer means chatbot. A chatbot is a product interface. A transformer is a model architecture.

Mental model

Think of a transformer as a factory line where each station refines token meaning using context from the whole sequence.

Transformers: questions and answer hints

Q1

Why did transformers beat many earlier sequence models?

ANSWER HINTS

- Parallel training was more efficient.
- Attention handled long-range context better.
- Scaling with data and compute worked unusually well.

Q2

What are the main building blocks of a transformer?

ANSWER HINTS

- Token embeddings, positional information, attention, feed-forward layers, normalization, and output head.
- Mention that details vary by architecture.
- Keep the answer conceptual unless asked for math.

Q3

How is a transformer different from an LLM?

ANSWER HINTS

- Transformer is architecture.
- LLM is a large model trained on language or code.
- Many LLMs use transformers, but the terms are not identical.

Answer pattern: define the concept, explain why it matters, give one concrete product example, then name one limitation or risk.

PART 02

Inside The Model

How LLMs produce useful answers.

06 LLMs

07 Context Window

08 Temperature

09 Hallucination

10 Prompt Engineering

LLMs

A large language model predicts likely token sequences based on patterns learned from massive training data.

Detailed explanation

- An LLM is usually trained on huge corpora of text and code. The base training objective is often next-token prediction.
- Because prediction requires grammar, facts, style, reasoning patterns, and code structure, useful capabilities emerge from the training process.
- A deployed assistant is more than a base LLM. It usually includes instruction tuning, safety layers, tools, memory, retrieval, and product logic.

Why it matters

- LLMs are the foundation for chatbots, copilots, agents, and AI search.
- They are powerful pattern engines, not guaranteed truth engines.
- They need product scaffolding to become reliable systems.

Common trap

Do not equate fluent language with verified knowledge or reasoning correctness.

Mental model

Think of an LLM as an autocomplete system that became broad enough to simulate many useful cognitive tasks.

LLMs: questions and answer hints

Q1

What is an LLM trained to do?

ANSWER HINTS

- Predict tokens based on context.
- Capabilities emerge because prediction is a rich task.
- Post-training makes it more assistant-like.

Q2

Why can LLMs answer questions if they are just predicting tokens?

ANSWER HINTS

- Training data contains many question-answer patterns.
- The model learns statistical structure and representations.
- Answering is generated as a likely continuation.

Q3

What should a team add around an LLM for production use?

ANSWER HINTS

- Retrieval, tools, evals, logging, guardrails, and fallback paths.
- Human review for high-risk workflows.
- Cost and latency controls.

Answer pattern: define the concept, explain why it matters, give one concrete product example, then name one limitation or risk.

Context Window

The context window is the maximum amount of input and output a model can consider at once.

Detailed explanation

- A model does not see an infinite conversation. It sees the tokens placed inside its current context window.
- The window includes system instructions, chat history, retrieved documents, tool results, and the answer being generated.
- Bigger context helps, but it does not guarantee perfect recall. Models can still miss details, overweight recent text, or struggle with long middle sections.

Why it matters

- It sets the working memory budget for AI workflows.
- It affects cost, latency, and answer quality.
- It forces teams to choose what context is truly relevant.

Common trap

Do not treat context window as permanent memory. It is temporary working context for one model call.

Mental model

Think of it as the model's desk space. You can place more papers on the desk, but clutter still hurts focus.

Context Window: questions and answer hints

Q1

What counts toward the context window?

ANSWER HINTS

- System prompt, user messages, previous turns, retrieved context, tool outputs, and generated response.
- Input and output share the budget.
- Hidden product instructions can also count.

Q2

Why is a bigger context window not always better?

ANSWER HINTS

- Higher cost and latency.
- More irrelevant context can confuse retrieval and reasoning.
- Needle-in-haystack performance varies.

Q3

How would you manage long documents?

ANSWER HINTS

- Chunk, retrieve, summarize, and cite relevant sections.
- Use hierarchy and metadata.
- Evaluate whether the model finds the needed facts.

Answer pattern: define the concept, explain why it matters, give one concrete product example, then name one limitation or risk.

Temperature

Temperature controls how deterministic or varied a model's sampling behavior is.

Detailed explanation

- When a model predicts the next token, it assigns probabilities to many possible tokens. Temperature changes how sharply or loosely those probabilities are sampled.
- Lower temperature makes outputs more predictable and conservative. Higher temperature increases variation, creativity, and risk of drift.
- Temperature is not intelligence. It is a generation control that should match the task.

Why it matters

- It is one of the simplest knobs for output behavior.
- It helps tune consistency versus diversity.
- It matters for code, facts, ideation, rewriting, and test generation.

Common trap

Do not use high temperature for tasks that need faithful extraction or compliance with exact facts.

Mental model

Think of temperature as choosing between a strict editor and a playful brainstorming partner.

Temperature: questions and answer hints

Q1

When would you use low temperature?

ANSWER HINTS

- Extraction, classification, code generation, and factual summaries.
- Anything that values repeatability.
- Pair with clear schemas and evals.

Q2

When would you use higher temperature?

ANSWER HINTS

- Brainstorming, creative writing, ad variations, naming, and exploration.
- When diversity is valuable.
- Still check quality afterward.

Q3

Does temperature fix hallucination?

ANSWER HINTS

- No.
- Lower temperature can reduce random drift but not guarantee truth.
- Use retrieval, grounding, and verification.

Answer pattern: define the concept, explain why it matters, give one concrete product example, then name one limitation or risk.

Hallucination

Hallucination happens when a model generates plausible text that is unsupported, false, or fabricated.

Detailed explanation

- LLMs are optimized to generate likely continuations, not to guarantee truth. If the prompt asks for information the model lacks, it may still produce a confident answer.
- Hallucinations can include fake citations, wrong numbers, invented APIs, non-existent policies, and misleading summaries.
- The fix is not a single prompt. Production systems reduce hallucination through retrieval, constrained outputs, citations, tool checks, evals, and human review where needed.

Why it matters

- It is the central reliability risk in many AI products.
- It affects trust, compliance, and user safety.
- It changes how teams design workflows and guardrails.

Common trap

Do not promise zero hallucinations. Design systems that detect, reduce, and contain them.

Mental model

Think of hallucination as confident pattern completion without enough evidence.

Hallucination: questions and answer hints

Q1

Why do LLMs hallucinate?

ANSWER HINTS

- They predict plausible tokens.
- They may lack grounding or current information.
- They do not automatically know when they are wrong.

Q2

How would you reduce hallucinations in an enterprise assistant?

ANSWER HINTS

- Ground answers in retrieved documents.
- Require citations and tool verification.
- Add refusal behavior, evals, and monitoring.

Q3

What is the difference between hallucination and uncertainty?

ANSWER HINTS

- Uncertainty is not knowing.
- Hallucination is presenting unsupported content as if true.
- Good systems expose uncertainty rather than hiding it.

Answer pattern: define the concept, explain why it matters, give one concrete product example, then name one limitation or risk.

Prompt Engineering

Prompt engineering is the practice of shaping instructions, context, and output format to get better model behavior.

Detailed explanation

- A prompt tells the model what role to take, what goal to pursue, what context to use, what constraints to follow, and what format to return.
- Good prompts reduce ambiguity. They include examples, definitions, acceptance criteria, and boundaries when the task needs precision.
- Prompting is useful, but it is not a substitute for product design, evals, data quality, or workflow architecture.

Why it matters

- It is the fastest way to improve many AI workflows.
- It helps non-engineers shape model behavior.
- It becomes a reusable product asset when paired with tests.

Common trap

Do not treat prompts as magic spells. Treat them as instructions that need evaluation and iteration.

Mental model

Think of a prompt like a creative brief plus operating procedure for the model.

Prompt Engineering: questions and answer hints

Q1

What makes a prompt good?

ANSWER HINTS

- Clear task, context, constraints, examples, and output format.
- Defines success and failure.
- Avoids conflicting instructions.

Q2

How would you improve a poor prompt?

ANSWER HINTS

- Add role, task, context, criteria, and examples.
- Specify output structure.
- Test across real cases.

Q3

Where does prompt engineering stop being enough?

ANSWER HINTS

- When the model lacks needed information.
- When deterministic systems or tools are required.
- When reliability needs evals and architecture.

Answer pattern: define the concept, explain why it matters, give one concrete product example, then name one limitation or risk.

PART 03

Making Models Useful

How base models become product-ready.

11 Transfer Learning

12 Fine Tuning

13 RLHF

14 LoRA

15 Quantization

Transfer Learning

Transfer learning reuses knowledge from a pretrained model for a new task.

Detailed explanation

- Instead of training a model from scratch, teams start with a model that has already learned broad patterns from large datasets.
- They then adapt it through prompting, retrieval, fine tuning, or smaller task-specific layers.
- Transfer learning is why startups and product teams can build powerful AI features without owning frontier-scale training infrastructure.

Why it matters

- It lowers cost and time to build AI systems.
- It makes specialized use cases feasible.
- It explains the value of foundation models.

Common trap

Do not assume transfer learning automatically solves domain accuracy. The target task still needs data, evaluation, and product constraints.

Mental model

Think of hiring a generalist who already speaks the language, then training them on your company playbook.

Transfer Learning: questions and answer hints

Q1

Why is transfer learning important?

ANSWER HINTS

- It avoids training from zero.
- It reuses broad representations.
- It makes domain adaptation cheaper.

Q2

How can you adapt a pretrained model?

ANSWER HINTS

- Prompting, RAG, fine tuning, adapters, or task-specific heads.
- Pick based on data, latency, cost, and control needs.
- Start with the simplest method that works.

Q3

What risks remain after transfer learning?

ANSWER HINTS

- Domain mismatch.
- Bias and gaps from source data.
- Overfitting or false confidence on the target task.

Answer pattern: define the concept, explain why it matters, give one concrete product example, then name one limitation or risk.

Fine Tuning

Fine tuning continues training a pretrained model on task-specific examples.

Detailed explanation

- Fine tuning changes model weights using curated examples. It can teach style, format, domain behavior, classification rules, or task-specific patterns.
- It is useful when prompting and retrieval are not enough, or when you need consistent behavior at scale.
- Fine tuning is not ideal for adding frequently changing facts. RAG is usually better for fresh knowledge.

Why it matters

- It can improve consistency and reduce prompt length.
- It can encode domain-specific behavior.
- It can lower latency or cost by moving instructions into weights.

Common trap

Do not fine tune to store fast-changing knowledge. Use retrieval or tools for that.

Mental model

Think of fine tuning as practicing with many examples until a behavior becomes natural.

Fine Tuning: questions and answer hints

Q1

When should you fine tune instead of prompt?

ANSWER HINTS

- When behavior must be consistent across many cases.
- When examples define the task better than instructions.
- When long prompts are costly or fragile.

Q2

What data do you need for fine tuning?

ANSWER HINTS

- High-quality input-output examples.
- Representative edge cases.
- Clear evaluation set separate from training data.

Q3

How is fine tuning different from RAG?

ANSWER HINTS

- Fine tuning changes weights.
- RAG supplies external context at request time.
- Fine tuning shapes behavior, RAG supplies knowledge.

Answer pattern: define the concept, explain why it matters, give one concrete product example, then name one limitation or risk.

RLHF

RLHF uses human preference feedback to make model outputs more helpful and aligned.

Detailed explanation

- Reinforcement Learning from Human Feedback starts with humans comparing model outputs and ranking which response is better.
- A reward model learns those preferences, then the model is optimized to produce outputs that score better under the reward model.
- RLHF helped make chat models more useful, polite, and instruction-following, but it can also introduce over-optimization and refusal quirks.

Why it matters

- It explains why chat models feel different from raw base models.
- It aligns output style with human preferences.
- It shows that model quality is partly a product and feedback problem.

Common trap

Do not say RLHF guarantees safety or truth. It optimizes for preferred outputs, which may still be wrong.

Mental model

Think of RLHF as coaching the model with human taste tests.

RLHF: questions and answer hints

Q1

What problem does RLHF solve?

ANSWER HINTS

- Base models are not naturally helpful assistants.
- Human preferences guide style and helpfulness.
- It improves instruction-following behavior.

Q2

What are the steps in RLHF?

ANSWER HINTS

- Collect comparisons.
- Train a reward model.
- Optimize the model against the reward signal.

Q3

What are limitations of RLHF?

ANSWER HINTS

- Human feedback can be biased or inconsistent.
- The model can optimize for sounding good.
- It does not replace factual grounding.

Answer pattern: define the concept, explain why it matters, give one concrete product example, then name one limitation or risk.

LoRA

LoRA adapts large models by training small low-rank adapter matrices instead of all weights.

Detailed explanation

- Low-Rank Adaptation freezes the base model and adds small trainable adapter layers.
- This makes fine tuning cheaper, faster, and easier to store or swap for different tasks.
- LoRA is widely used in open-source LLM and image model workflows because it lets teams customize models without retraining everything.

Why it matters

- It reduces compute and memory needs.
- It enables many specialized adapters on one base model.
- It makes experimentation more accessible.

Common trap

Do not describe LoRA as a separate model from scratch. It is an adaptation technique layered onto a base model.

Mental model

Think of LoRA as adding a lightweight specialist module to a general engine.

LoRA: questions and answer hints

Q1

Why is LoRA efficient?

ANSWER HINTS

- It freezes most weights.
- It trains small low-rank matrices.
- It stores task-specific changes compactly.

Q2

When would LoRA be useful?

ANSWER HINTS

- Domain style adaptation.
- Multiple customer-specific variants.
- Resource-constrained fine tuning.

Q3

What tradeoffs does LoRA have?

ANSWER HINTS

- It may be less powerful than full fine tuning for some tasks.
- Adapter quality depends on data.
- Managing many adapters adds operational complexity.

Answer pattern: define the concept, explain why it matters, give one concrete product example, then name one limitation or risk.

Quantization

Quantization reduces numeric precision so models use less memory and run faster.

Detailed explanation

- Model weights are usually stored as numbers. Quantization represents those numbers with fewer bits, such as 8-bit or 4-bit precision.
- Lower precision reduces memory, bandwidth, and sometimes latency. This makes local and edge deployment more practical.
- The tradeoff is possible quality loss. Good quantization methods minimize that loss for the target hardware and task.

Why it matters

- It helps run larger models on smaller devices.
- It lowers serving costs.
- It matters for privacy, offline use, and latency-sensitive products.

Common trap

Do not assume quantization is always free. Measure quality, speed, and stability after compression.

Mental model

Think of quantization as saving a high-resolution image at a smaller file size while trying to preserve what matters.

Quantization: questions and answer hints

Q1

Why quantize a model?

ANSWER HINTS

- Reduce memory footprint.
- Improve deployment feasibility.
- Lower cost or latency.

Q2

What can go wrong after quantization?

ANSWER HINTS

- Accuracy can drop.
- Rare tasks may degrade more than common ones.
- Some hardware may not benefit as expected.

Q3

How would you evaluate a quantized model?

ANSWER HINTS

- Compare task quality against baseline.
- Measure latency, throughput, memory, and cost.
- Test real production prompts and edge cases.

Answer pattern: define the concept, explain why it matters, give one concrete product example, then name one limitation or risk.

PART 04

Real Systems

How AI is shipped in the wild.

16 RAG

17 Vector Databases

18 AI Agents

19 Chain of Thought

20 Diffusion Models

RAG

RAG retrieves relevant external information and gives it to the model before generation.

Detailed explanation

- Retrieval Augmented Generation combines search with generation. The system retrieves relevant documents, chunks, or records, then asks the model to answer using that context.
- RAG is useful when knowledge changes, when answers need citations, or when the model should use private company data.
- A good RAG system depends on chunking, embeddings, ranking, metadata filters, prompt design, citation handling, and evaluation.

Why it matters

- It reduces unsupported answers.
- It connects LLMs to private or current knowledge.
- It makes answers more auditable.

Common trap

Do not assume adding a vector database automatically creates good RAG. Retrieval quality is the product.

Mental model

Think of RAG as an open-book exam where the model gets the relevant pages before answering.

RAG: questions and answer hints

Q1

When should you use RAG?

ANSWER HINTS

- Private knowledge, current facts, citations, or large document bases.
- When fine tuning would be too static.
- When users need source-grounded answers.

Q2

What are the main parts of a RAG pipeline?

ANSWER HINTS

- Ingestion, chunking, embeddings, storage, retrieval, reranking, prompting, generation, and citations.
- Add evals and monitoring.
- Include permissions when data is sensitive.

Q3

Why can RAG still fail?

ANSWER HINTS

- Bad chunks, poor retrieval, stale data, missing permissions, or weak prompts.
- The model can ignore context.
- Evaluation must test retrieval and generation separately.

Answer pattern: define the concept, explain why it matters, give one concrete product example, then name one limitation or risk.

Vector Databases

Vector databases store embeddings and retrieve items by semantic similarity.

Detailed explanation

- A vector database indexes embeddings so systems can quickly find nearby vectors for a query embedding.
- It is often used in RAG, recommendations, semantic search, duplicate detection, and clustering.
- Production vector search also needs metadata filtering, access control, reranking, freshness, and observability.

Why it matters

- It powers search by meaning.
- It helps AI systems find relevant context at scale.
- It turns embeddings into useful infrastructure.

Common trap

Do not use vector search alone for everything. Exact keyword filters and structured metadata still matter.

Mental model

Think of it as a library where books are shelved by meaning rather than alphabetically.

Vector Databases: questions and answer hints

Q1

What does a vector database store?

ANSWER HINTS

- Embeddings plus metadata and document references.
- The original text may be stored separately or alongside vectors.
- Indexes make nearest-neighbor search fast.

Q2

How do you improve retrieval quality?

ANSWER HINTS

- Better chunking, metadata filters, hybrid search, reranking, and evals.
- Tune embedding model for the domain.
- Track failed queries.

Q3

When is keyword search better?

ANSWER HINTS

- Exact IDs, SKUs, error codes, legal citations, and names.
- Hybrid search often beats either method alone.
- Use the retrieval mode that matches the user intent.

Answer pattern: define the concept, explain why it matters, give one concrete product example, then name one limitation or risk.

AI Agents

AI agents use models to plan, call tools, observe results, and continue toward a goal.

Detailed explanation

- An agent is not just a chatbot. It can decide what action to take, use tools such as search or code execution, inspect results, and adapt.
- Agentic systems need constraints because autonomous tool use creates risk. Good agents have scoped tools, permissions, budgets, logging, and recovery paths.
- The most reliable agents often combine model reasoning with deterministic workflows and human approval at important boundaries.

Why it matters

- Agents move AI from answering to doing.
- They enable multi-step workflows.
- They introduce new product, security, and reliability requirements.

Common trap

Do not call every LLM workflow an agent. Tool use, goal pursuit, and iterative control matter.

Mental model

Think of an agent as an intern with tools, a checklist, and supervision.

AI Agents: questions and answer hints

Q1

What makes an AI system an agent?

ANSWER HINTS

- Goal-directed behavior.
- Tool use and observation.
- Multi-step planning or control loop.

Q2

What guardrails would you add to an agent?

ANSWER HINTS

- Tool permissions, budgets, audit logs, approvals, sandboxing, and rollback.
- Clear stop conditions.
- Monitoring for loops and unsafe actions.

Q3

Where do agents work best today?

ANSWER HINTS

- Well-scoped workflows.
- Tasks with clear success criteria.
- Environments where actions can be checked before execution.

Answer pattern: define the concept, explain why it matters, give one concrete product example, then name one limitation or risk.

Chain of Thought

Chain-of-thought style prompting encourages a model to break a problem into intermediate reasoning steps.

Detailed explanation

- For hard tasks, models often perform better when the problem is decomposed into steps before giving a final answer.
- In products, you usually do not need to show the full hidden reasoning. You can ask for concise rationale, checks, or structured intermediate outputs instead.
- Reasoning prompts help, but they are not proof. Math, logic, and factual claims still need verification.

Why it matters

- It improves performance on multi-step tasks.
- It encourages decomposition and self-checking.
- It influences how teams design reasoning workflows.

Common trap

Do not treat chain-of-thought text as guaranteed faithful reasoning. It can be plausible post-hoc explanation.

Mental model

Think of it as asking someone to use scratch paper before giving the final answer.

Chain of Thought: questions and answer hints

Q1

Why can step-by-step prompting improve answers?

ANSWER HINTS

- It decomposes the task.
- It gives the model room to track intermediate variables.
- It can reduce shortcut mistakes.

Q2

Should products expose full chain of thought?

ANSWER HINTS

- Usually no.
- Show concise rationale, assumptions, checks, or citations instead.
- Protect private reasoning and avoid misleading users.

Q3

How would you verify model reasoning?

ANSWER HINTS

- Use tools, tests, calculators, code execution, or external sources.
- Ask for final answer plus verifiable evidence.
- Evaluate on known benchmark cases.

Answer pattern: define the concept, explain why it matters, give one concrete product example, then name one limitation or risk.

Diffusion Models

Diffusion models learn to reverse noise into structured media such as images, audio, or video.

Detailed explanation

- During training, noise is gradually added to data. The model learns how to reverse that process and recover structure from noisy inputs.
- At generation time, the model starts from noise and repeatedly denoises it while guided by a prompt, image, or other condition.
- Diffusion powers many image and video systems. It is strong at rich media synthesis but can struggle with exact text, geometry, identity consistency, and controllability.

Why it matters

- It explains modern image and video generation.
- It shows a different generative approach than next-token prediction.
- It matters for creative tools, advertising, design, and synthetic media.

Common trap

Do not assume diffusion models reason about images like humans. They learn a denoising process guided by training data and conditioning.

Mental model

Think of it as sculpting a clear image out of static, one cleanup pass at a time.

Diffusion Models: questions and answer hints

Q1

How does a diffusion model generate an image?

ANSWER HINTS

- Start from noise.
- Iteratively denoise.
- Use prompt or conditioning to steer the result.

Q2

How is diffusion different from an LLM?

ANSWER HINTS

- Diffusion iteratively denoises continuous media.
- LLMs usually generate discrete tokens sequentially.
- Both are generative, but the mechanics differ.

Q3

What product risks come with diffusion models?

ANSWER HINTS

- Copyright, identity misuse, safety, brand consistency, and factual visual claims.
- Need review workflows and content policy.
- Need controls for repeatability and provenance.

Answer pattern: define the concept, explain why it matters, give one concrete product example, then name one limitation or risk.

FINAL CHECKLIST

Before an AI interview, make sure you can do these five things.

Define simply

Explain each concept without jargon in 30 to 60 seconds.

Compare choices

Know when to use prompting, RAG, fine tuning, tools, or human review.

Name failure modes

Hallucination, bad retrieval, stale context, poor evals, and unsafe tool use.

Think like a PM

Connect each concept to user value, cost, latency, risk, and measurement.

Use examples

Ground every answer in a real workflow, not just a definition.

WHO MADE THIS DECK

Vishal Jaiswal

AI and Analytics Leader. 15+ years. Germany / India.

I build practical AI systems for operators, PMs, analysts, and founders who want to understand the concepts well enough to ship with them.

20
AI concepts explained

60
Interview prompts and hints

47+
Study slides and checklists

`jaiswal-vishal.vercel.app`

`linkedin.com/in/vishal-jaiswal-analytics-leader`

YOU HAVE THE FULL PREP DECK

Now turn this into *FLUENCY*

Use the deck to practice crisp explanations, sharper interview answers, and better AI product judgment.

01

Read one section

Do not skim all 20 concepts in one sitting.

02

Answer out loud

Use the interview slides before looking at hints.

03

Apply to work

Map each concept to a product, workflow, or risk.

Follow Vishal Jaiswal for practical AI systems, PM workflows, and operator playbooks.