Claude Certified Architect

FOUNDATIONS EXAM

Study Guide

A complete question bank organized by the five official exam domains — with the correct answer and a full explanation for every question.

D1	D2	D3	D4	D5
27%	18%	20%	20%	15%

77 questions

5 domains · answers & explanations · searchable text

CONTENTS

Five domains · 77 questions

Each part maps to an official exam domain. Weight = share of scored content.

01 **Part 1 · Agentic Architecture & Orchestration** — 22 questions
Multi-agent orchestration, coordinator–subagent patterns, task decomposition and parallel execution.

02 **Part 2 · Tool Design & MCP Integration** — 20 questions
Tool design, built-in tools (Grep / Glob / Bash) and MCP servers, tools and resources.

03 **Part 3 · Claude Code Configuration & Workflows** — 18 questions
Claude Code flags, CLAUDE.md, sessions, permissions and developer workflows.

04 **Part 4 · Prompt Engineering & Structured Output** — 14 questions
Prompt engineering, few-shot examples, structured output, confidence and severity metadata.

05 **Part 5 · Context Management & Reliability** — 3 questions
Context management, batch / async APIs, retries and reliability.

01

DOMAIN 1

Agentic Architecture & Orchestration

22 questions

27% of scored content

Multi-agent orchestration, coordinator–subagent patterns, task decomposition and parallel execution.

1 Your pipeline reviews every PR using a single API call with a static prompt containing the diff and full text of each changed file — unchanged files are not included. Reviews are posted asynchronously and don't block PR creation. Developers report that reviews consistently miss bugs involving cross-file interactions — for example, a PR renames a function's parameters but the review doesn't flag callers in unchanged files that still use the old argument order. Evaluation shows cross-file bugs account for 35% of production incidents from reviewed PRs. What is the most effective change to your review design?

D1 · Agentic Architecture & Orchestration

A Redesign the review as a turn-limited agentic task where the model can read files and search the codebase via tools, following references to verify cross-file findings.

B Add chain-of-thought instructions asking the model to list all external references in the diff, then reason step-by-step about how each change might affect callers in other files.

C Run parallel review passes per changed file with direct dependents included in each pass, then aggregate and deduplicate findings using a final summarization call.

D Use static analysis to build a dependency graph of changed code, then expand the prompt to include all files within two dependency hops of any changed file.

Correct answer: A

EXPLANATION

The failure is a missing-context problem: callers live in unchanged files the static prompt never includes, so chain-of-thought (B) cannot help the model reason about code it cannot see, and static heuristics (C, D) bloat context while still missing dynamic or multi-hop references. Anthropic's agent guidance favors agentic search — letting the model retrieve exactly the context it needs via file-read/search tools — and the asynchronous, non-blocking review pipeline tolerates the added latency, while a turn limit bounds cost.

2 An engineer asks your agent to add comprehensive tests to a legacy codebase with 200 files and minimal existing test coverage. The engineer hasn't specified which modules to prioritize. How should the agent decompose this open-ended task?

D1 · Agentic Architecture & Orchestration

A. Systematically read all 200 files to create a complete function inventory before writing any tests, ensuring the testing plan accounts for every function before beginning.

B. Create a fixed testing schedule upfront based on directory structure, allocating equal effort to each top-level directory regardless of code complexity or business importance.

C. Start writing tests for the first module alphabetically, using test failures and imports to discover related files organically.

D. Use Glob and Grep to map codebase structure, identify heavily-coupled modules, create a prioritized plan for high-impact areas, and revise as dependencies are discovered.

Correct answer: D

EXPLANATION

Anthropic's agent best practices favor efficient context gathering with search tools (Glob/Grep) over exhaustively reading every file, which would waste context and time. Prioritizing high-impact, heavily-coupled modules and iteratively revising the plan as dependencies surface is the correct way to decompose an open-ended task; A is wasteful, B ignores impact/complexity, and C is arbitrary and unplanned.

3

An engineer submits two requests: • Request A: "Rename the getUserData function to fetchUserProfile everywhere it's used." • Request B: "Improve error handling throughout the data processing module—add try/catch blocks, meaningful error messages, and ensure failures don't silently corrupt data." For which request does specifying an explicit multi-phase workflow (such as analyze propose implement with review) most improve outcome quality?

D1 · Agentic Architecture & Orchestration

A Request B, the error handling task

B Both requests benefit equally

C Request A, the function rename task

D Neither request benefits significantly

Correct answer: A

EXPLANATION

Request B is open-ended and judgment-heavy — "improve error handling" requires analyzing current behavior, deciding where try/catch belongs, and choosing meaningful messages, so an explicit analyze propose implement-with-review workflow materially raises quality and catches bad design before code changes. Request A is a mechanical, well-defined rename that a simple find-and-replace-style execution handles fine, so the multi-phase structure adds little.

4 Your code review assistant needs to analyze pull requests and provide feedback on three aspects: code style compliance, potential security issues, and documentation completeness. Each aspect requires reading files, running analysis tools, and generating a report section. The review process follows the same three-step workflow for every PR. Which task decomposition pattern is most appropriate for this workflow?

D1 · Agentic Architecture & Orchestration

A. Single comprehensive prompt—include all instructions in one prompt and let the model handle all three aspects simultaneously.

B. Routing—classify each PR by type (feature, bugfix, refactor) first, then route to different review prompts optimized for that category.

C. Prompt chaining—break the review into sequential steps where each aspect (style, security, documentation) is analyzed separately, with outputs combined in a final synthesis step.

D. Orchestrator-workers—have a central LLM analyze each PR to dynamically determine which checks are needed, then delegate to specialized worker LLMs for each identified subtask.

Correct answer: C

EXPLANATION

The scenario explicitly states the review "follows the same three-step workflow for every PR," which per Anthropic's "Building Effective Agents" guidance is the textbook case for the prompt chaining workflow: a fixed, predictable sequence of subtasks each handled by a focused prompt, then synthesized. Orchestrator-workers (D) is only warranted when the needed subtasks vary unpredictably per input, and routing (B) applies when inputs fall into distinct categories needing different handling—neither fits a fixed, identical workflow.

5 Your multi-agent research pipeline crashed after processing 12 of 28 documents. The web search agent had identified relevant sources, the document analyzer had partially completed extraction, and the synthesizer had begun pattern identification. You need to resume processing without repeating work or losing fidelity of prior findings. What state management approach best balances information fidelity with context efficiency when restoring agent state?

D1 · Agentic Architecture & Orchestration

A. Have each agent maintain its own persistent state file and reload it independently at the start of each session.

B. Persist the coordinator's conversation log containing all task delegations and responses, providing this to agents when resuming.

C. Index all agent outputs in a shared vector store. When resuming, each agent queries the store using semantic search to retrieve relevant prior findings.

D. Have each agent persist a structured export to a known location. On resume, the coordinator loads the manifest and injects relevant state into agent prompts.

Correct answer: D

EXPLANATION

Anthropic's multi-agent best practices recommend persisting structured artifacts to external storage and having the orchestrator selectively re-inject only the relevant state into each subagent's context. This preserves fidelity (lossless structured exports plus a manifest of completed work) while staying context-efficient, whereas A lacks coordinated resume logic, B floods agents with an entire conversation log, and C's semantic search is lossy and may miss or garble prior findings.

6

Analysis reveals the coordinator invokes the web search subagent, waits for its response, then invokes the document analysis subagent and waits again. These tasks are independent—neither requires the other's output. How should you modify the system to run these subagents concurrently?

D1 · Agentic Architecture & Orchestration

Scenario: Multi-Agent Research — Production monitoring shows the research phase takes longer than expected.

A. Switch both subagents to use a Haiku-tier model instead of Sonnet to reduce their individual execution time.

B. Create an async orchestration layer outside the agent that spawns parallel threads, each running a separate coordinator-subagent pair, then aggregates results.

C. Structure the coordinator to emit both Task tool calls (for web search and document analysis) in a single response message rather than across separate conversation turns.

D. Add detailed instructions to the coordinator's system prompt explaining the performance benefits of parallel execution and requesting it invoke both subagents at the same time.

Correct answer: C

EXPLANATION

In Claude's agent architecture, subagents run concurrently when the coordinator emits multiple Task tool calls in a single assistant message — the harness executes parallel tool_use blocks from the same turn simultaneously. Option A only speeds up each serial step, B adds unnecessary external infrastructure duplicating what the agent loop already supports, and D merely explains/requests parallelism without ensuring the structural mechanism (same-message tool calls) that actually produces concurrent execution.

7 The coordinator agent has AgentDefinitions configured for all four specialized subagents, each with appropriate descriptions, prompts, and tool restrictions. During testing, you notice the coordinator correctly reasons about when to delegate—it generates messages like "I'll ask the web search agent to find sources on this topic"—but no subagent execution ever occurs. The coordinator then proceeds as if the delegation happened and continues with incomplete information. Logs show no errors. What is the most likely cause?

D1 · Agentic Architecture & Orchestration

A. The AgentDefinitions are configured correctly, but the coordinator's system prompt doesn't explicitly list the available subagent types, preventing the model from knowing they can be invoked.

B. Subagent context isolation means task descriptions from the coordinator don't automatically reach subagents; you need to configure explicit context forwarding in ClaudeAgentOptions.

C. The coordinator's max_tokens setting is too low, causing the Task tool invocation to be truncated before the subagent type parameter can be specified.

D. The coordinator's allowedTools configuration doesn't include "Task", so while it can reason about delegation, it cannot invoke the tool required to spawn subagents.

Correct answer: D

EXPLANATION

In the Claude Agent SDK, subagents defined via AgentDefinitions are launched through the Task tool; if allowedTools omits "Task", the coordinator can still verbally reason about delegating but has no mechanism to actually spawn subagents, and it silently continues without them—matching the symptom of no execution and no errors. Truncated max_tokens (C) or missing prompt listings (A) would produce errors or different behavior, and context isolation (B) doesn't prevent invocation.

8 A landmark case citing 12 precedents takes over 3 minutes to analyze completely. What's the most effective way to reduce this latency while preserving the coordinator's ability to monitor and debug the system?

D1 · Agentic Architecture & Orchestration

Scenario: Multi-Agent Research — When analyzing complex legal cases that cite multiple precedents, the document analysis subagent processes each sequentially.

- **A** Have the coordinator spawn parallel document analysis subagents, each handling a subset of precedents, then aggregate results before synthesis
- **B** Enable the document analysis subagent to spawn its own specialized subagents dynamically when it encounters cases with many citations
- **C** Create a recursive agent hierarchy where analysis agents subdivide work among child agents until reaching single-precedent granularity
- **D** Implement a message queue where precedent analysis tasks are processed asynchronously by a pool of worker agents

Correct answer: A

EXPLANATION

Anthropic's recommended multi-agent design is the orchestrator-worker pattern with a flat hierarchy: the coordinator itself spawns parallel subagents and aggregates their results, which cuts latency through parallelism while keeping all spawning, monitoring, and debugging centralized in one place. Options B and C create nested/recursive agent hierarchies that severely degrade observability and debuggability, and D introduces asynchronous queue infrastructure that decouples work from the coordinator's direct visibility.

9

Investigation shows that while the web search and document analysis agents correctly attach citations to their outputs, the synthesis agent loses track of which sources support which conclusions when combining findings. What's the most effective architectural change?

D1 · Agentic Architecture & Orchestration

Scenario: Multi-Agent Research — In production, final reports frequently contain claims without proper source attribution.

A. Have the coordinator inject source identifier prefixes into text before each handoff, then parse these prefixes at report generation to reconstruct citations.

B. Maintain complete transcripts of all subagent interactions and add a citation-resolution agent to analyze logs and determine attributions before report generation.

C. Require all subagents to output structured claim-source mappings that the synthesis agent must preserve and merge when combining findings from multiple sources.

D. Add a verification step where the report generator uses semantic similarity matching against original sources to reconstruct which claims came from which documents.

Correct answer: C

EXPLANATION

The failure is information loss at the synthesis stage, so the fix is to carry attribution as structured data end-to-end: subagents emit explicit claim-source mappings and the synthesis agent is required to preserve and merge them. This matches Anthropic's multi-agent best practice of structured outputs over fragile text-prefix conventions (A), costly post-hoc log analysis (B), or unreliable semantic-similarity reconstruction (D).

10 Users report that final reports sometimes lack depth on specific subtopics. Investigation shows that the document analysis agent frequently identifies gaps—for instance, noting "the retrieved sources discuss API authentication but lack details on token refresh patterns"—but under the current strict pipeline, this insight isn't actionable since search has already completed. What's the most effective architectural change?

D1 · Agentic Architecture & Orchestration

A. Have the coordinator review analysis output for gap indicators and re-invoke search with gap-informed queries when gaps are detected.

B. Have the synthesis agent attach confidence scores to each section and flag areas with insufficient coverage for manual review.

C. Add a research planning agent before the search phase that decomposes topics into specific sub- questions.

D. Have the analysis agent report specific gaps to the coordinator, which triggers targeted searches and re-invokes analysis until sufficient.

Correct answer: D

EXPLANATION

The core problem is a rigid one-way pipeline where gap insights arrive after search is finished, so the fix is a closed feedback loop: the analysis agent surfaces its specific gaps as structured output to the coordinator, which triggers targeted follow-up searches and re-runs analysis iteratively until coverage is sufficient. This matches Anthropic's orchestrator-worker agent pattern of iterating until a verifiable success criterion is met; option A is a weaker partial loop (coordinator inferring gaps from output, no re-analysis until sufficiency), while B and C don't make the gap insight actionable at runtime.

11 After the web search and document analysis subagents complete their tasks, the coordinator needs to spawn the synthesis subagent to synthesize the findings. What is the correct approach for providing the synthesis subagent with the information it needs?

D1 · Agentic Architecture & Orchestration

Scenario: Multi-Agent Research System.

A. Provide the subagent with tool definitions that allow it to request outputs from other subagents via callbacks

B. Include the complete findings from both subagents directly in the synthesis subagent's prompt

C. Pass reference identifiers and configure the subagent with read access to a shared memory store where other subagents deposited their results

D. Spawn the subagent with only a brief task description, relying on automatic context inheritance from the coordinator

Correct answer: C

EXPLANATION

Anthropic's multi-agent best practice is for subagents to deposit their full outputs as artifacts in external/shared storage and pass lightweight references, so the synthesis subagent reads the prior results directly rather than receiving lossy or token-heavy copies through prompts. Callbacks between subagents (A) and automatic context inheritance (D) don't exist in this architecture, and inlining complete findings into the prompt (B) bloats context and risks information loss.

12 A user is expanding the research system beyond its single web search agent by adding specialized data sources. They add a financial API agent that returns structured JSON with revenue, margins, and growth rates; a news monitoring agent that returns prose summaries of recent developments; and a patent analysis agent that returns structured lists of technology areas. The synthesis agent combines these into executive briefings. Currently, it converts everything to bullet points, causing financial comparisons to lose tabular clarity and news summaries to lose narrative flow. What change would most improve briefing quality?

D1 · Agentic Architecture & Orchestration

A. Add a format conversion layer between subagents and synthesis that transforms all outputs to a common intermediate representation

B. Update the synthesis agent to render each content type appropriately—financial data as tables, news as prose

C. Standardize all subagent outputs to prose summaries with inline citations

D. Standardize all subagent outputs to JSON with fields for claim, evidence, source, and confidence

Correct answer: B

EXPLANATION

The quality loss comes from flattening heterogeneous content into one uniform format (bullets); options A, C, and D just repeat that mistake with a different uniform representation. The best fix is to make the synthesis agent format-aware, rendering each content type in its natural form—tables for financial comparisons, prose for news narratives—preserving the strengths of each subagent's output.

13 g., "What year was the Paris Climate Agreement signed?") traverse all four subagents sequentially, consuming 40+ seconds and significant tokens per query. Complex comparative research benefits from the full pipeline. Your query distribution is diverse and evolving as users discover new applications. What's the most effective approach to optimize for varying query complexity?

D1 · Agentic Architecture & Orchestration

Scenario: Multi-Agent Research — In production, you observe that simple fact-checking queries (e.

A Create a fast-path for factual questions that bypasses subagents entirely, routing all other queries through the complete pipeline to ensure research thoroughness.

B Implement pattern-based routing that categorizes queries by structure (single-fact vs. comparative vs. analytical) and maps each category to a predefined subagent combination.

C Train a query complexity classifier on labeled historical data to predict optimal subagent combinations, retraining periodically as query patterns evolve.

D Have the coordinator analyze each query and dynamically decide which subagents to invoke based on its assessment of query requirements.

Correct answer: D

EXPLANATION

Anthropic's multi-agent research guidance recommends having the lead/orchestrator agent scale effort to query complexity, dynamically deciding how many and which subagents to invoke per query. Static fast-paths (A), hand-coded pattern routing (B), and a trained classifier needing labeled data and retraining (C) all break down as the query distribution is diverse and evolving, whereas the coordinator's LLM-based judgment adapts naturally without maintenance.

14 The coordinator provides detailed step-by-step instructions to the web search subagent, specifying exact search queries, source priorities, and date filters. Production monitoring reveals three issues: (1) the subagent reports "insufficient results" rather than trying alternative approaches when pre-specified searches fail, (2) research quality drops for emerging topics that don't match expected patterns, and (3) the subagent rarely surfaces valuable tangential sources. What's the most effective way to improve subagent adaptability?

D1 · Agentic Architecture & Orchestration

A Specify research goals and quality criteria (coverage breadth, source diversity, recency) rather than procedural steps, letting the subagent determine its search strategy.

B Remove procedural details entirely, delegating with simple goals like "research X thoroughly" and relying on the subagent's general capabilities.

C Add explicit fallback directives to the detailed instructions: "If specified searches yield fewer than N results, attempt alternative query formulations before reporting failure."

D Implement a topic classification step where the coordinator categorizes requests as "well-defined" or "exploratory" and uses different instruction styles for each category.

Correct answer: A

EXPLANATION

Anthropic's agent and multi-agent research guidance recommends delegating with clear objectives and success/quality criteria rather than prescriptive procedural scripts, letting the capable subagent adapt its own search strategy — which directly fixes all three observed failures. Option B removes necessary guidance entirely (under- specified delegation is a known failure mode), C patches only failure (1), and D adds coordinator complexity while keeping the brittle scripted approach.

15 The document analysis agent has a single analyze_document tool that takes a document and a free-text instruction parameter. During evaluation, requests like "extract the key financial metrics" often return narrative summaries, while "summarize the methodology" sometimes returns raw data tables. The synthesis agent reports that 35% of analysis results require re-requests with clarified instructions. What's the most effective way to improve reliability?

D1 · Agent Architecture & Orchestration

- **A** Split the generic tool into purpose-specific tools — extract_data_points, summarize_content, verify_claim_against_source — each with defined input/output contracts
- **B** Have the coordinator pre-classify each analysis request before passing instructions to the document analysis agent
- **C** Keep the single tool but add an analysis_type enum parameter requiring explicit selection between extraction, summarization, and verification modes
- **D** Enhance the tool description with detailed examples showing how different instruction phrasings should map to different output formats

Correct answer: A

EXPLANATION

Anthropic's tool-design guidance for agents recommends narrowly-scoped, purpose-specific tools with explicit input/output contracts rather than one generic free-text tool. The failures here are output-format ambiguity across distinct task types (extraction vs. summarization vs. verification), which splitting into dedicated tools with defined contracts fixes structurally; an enum (C) or better descriptions (D) still leave a single loosely-specified output contract, and pre-classification (B) doesn't change the unreliable tool interface.

16 After the web search agent and document analysis agent complete their tasks, the coordinator invokes the synthesis agent. However, the synthesis agent responds that it cannot complete the task because no research findings were provided. What is the most likely cause of this issue?

D1 · Agentic Architecture & Orchestration

A. The synthesis agent's context window is not large enough to hold the combined outputs from both previous agents.

B. The synthesis agent needs tools that can fetch results directly from the other agents' conversation histories.

C. The subagents need to share a single API connection to enable automatic context sharing between invocations.

D. The coordinator did not include the outputs from the previous agents in the synthesis agent's prompt.

Correct answer: D

EXPLANATION

In a coordinator/orchestrator multi-agent pattern, each subagent call is a stateless API invocation with its own isolated context — there is no automatic context sharing between agents. The coordinator must explicitly inject the previous agents' outputs into the synthesis agent's prompt; failing to do so produces exactly this "no research findings were provided" response. Options A–C are wrong: a context-window overflow would cause a different error, and neither cross-agent history-fetching tools nor a shared API connection are how context is passed between agents.

17 Production monitoring shows that follow-up queries like "summarize what we learned about market trends" consistently take 40+ seconds. Investigation reveals the coordinator spawns the synthesis subagent for each summarization request, passing 80K+ tokens of accumulated findings. The coordinator already has these findings in its context from orchestrating the research. What's the most effective way to improve response time for these follow-up summaries?

D1 · Agentic Architecture & Orchestration

Scenario: Multi-Agent Research System.

A Pre-generate and cache summaries at multiple granularities whenever new findings accumulate.

B Enable prompt caching on the synthesis subagent to reduce the overhead of repeatedly transferring the same research findings.

C Have the coordinator handle straightforward summarization requests directly using its existing context, reserving subagent spawning for complex analytical tasks.

D Spawn the synthesis subagent with reduced context and have it request specific findings from the coordinator on-demand.

Correct answer: C

EXPLANATION

The coordinator already has all the findings in its context, so spawning a subagent and re-transferring 80K+ tokens per summary is pure latency overhead. Anthropic's multi-agent best practice is to delegate to subagents only when separate context windows or parallelism add value; simple summarization should be handled directly by the orchestrator, eliminating both the spawn cost and the token transfer.

18 The synthesis agent receives summarized findings from the web search and document analysis agents, then passes a consolidated summary to the report generator. During testing, you discover the generated reports make factual claims without proper citations—the report generator cannot attribute statements to their original sources because that metadata was lost during the summarization steps. What's the most effective approach to ensure proper source attribution in the final reports?

D1 · Agentic Architecture & Orchestration

A. Instruct the synthesis agent to embed source references inline within its summary text using a consistent citation format.

B. Have the report generator query the web search agent to re-locate sources for claims in the final report.

C. Skip summarization and pass full raw outputs from web search and document analysis directly to the report generator.

D. Have each agent output structured data separating content summaries from source metadata (URLs, document names, page numbers).

Correct answer: D

EXPLANATION

Structured outputs that explicitly separate summarized content from source metadata (URLs, document names, page numbers) guarantee provenance survives every handoff in the pipeline, which is the Anthropic-recommended pattern for multi-agent data flow. Inline citations in free text (A) are easily lost or mangled during downstream summarization, re-locating sources after the fact (B) is unreliable and can mis-attribute claims, and passing raw outputs (C) defeats the purpose of summarization and bloats the report generator's context.

19 After the web search agent finds 25 sources (120K tokens of raw content), the document analysis agent extracts key insights (15K tokens), and the synthesis agent produces a coherent narrative draft (3K tokens), the coordinator must pass context to the report generation agent for the final output with proper source citations. What context-passing strategy provides the best balance of completeness and efficiency?

D1 · Agentic Architecture & Orchestration

A. Pass only the synthesis draft and have a separate post-processing pipeline match claims to sources and insert citations after the report is generated.

B. Pass the synthesis draft along with a structured source index that maps key claims to their source URLs and relevant excerpts.

C. Pass a condensed summary of all prior stages that preserves the main findings and attributes them to sources by name only.

D. Pass the full accumulated context from all prior agents.

Correct answer: B

EXPLANATION

Why it's correct: B. Correct. The synthesis gives the narrative; the source index gives the report generator exactly the binding it needs to cite without re-reading 120K tokens of raw content.

Option breakdown:

- **A:** Post-hoc matching is fragile — without the mapping the model used, you can't reliably bind claims to the right source. Hallucinated or misattributed citations are the usual failure mode.
- **B:** Correct. The synthesis gives the narrative; the source index gives the report generator exactly the binding it needs to cite without re-reading 120K tokens of raw content.
- **C:** Name-only attribution loses URLs and excerpts, so the report generator can't quote or verify — and 'by name only' tends to drift into vague citations.
- **D:** Maximum completeness but wasteful — 120K+ tokens of raw search content is mostly irrelevant noise at the report-generation stage.

20 The web search agent has gathered several relevant sources for a research topic. The document analysis agent now needs to examine these sources. How does information typically flow between these two specialized subagents?

D1 · Agentic Architecture & Orchestration

A. The agents communicate through an event-driven message queue, with the document analysis agent subscribing to web search completion events.

B. The web search agent directly invokes the document analysis agent, passing the discovered sources as parameters.

C. The coordinator agent receives the web search agent's output and includes relevant findings in the prompt when invoking the document analysis agent.

D. Both agents access a shared memory store where the web search agent writes findings and the document analysis agent reads them.

Correct answer: C

EXPLANATION

Why it's correct: C. Correct. In an orchestrator-worker pattern the coordinator is the hub. It collects each subagent's output and explicitly forwards the relevant parts into the next subagent's prompt.

Option breakdown:

- **A:** Event buses aren't part of the Claude subagent model. Subagents don't publish/subscribe to each other directly.
- **B:** Subagents are isolated — they can't directly call sibling subagents. That would also tightly couple them and defeat the orchestrator pattern.
- **C:** Correct. In an orchestrator-worker pattern the coordinator is the hub. It collects each subagent's output and explicitly forwards the relevant parts into the next subagent's prompt.
- **D:** A shared store can be layered in as an optimization, but it's not how subagents typically communicate — and it introduces stale-read and consistency problems.

21 When researching "renewable energy adoption," the web search agent returns recent statistics (2024: 35% adoption) while the document analysis agent extracts data from internal reports (2022: 18% adoption). The synthesis agent incorrectly flags these as contradictory sources rather than recognizing the data shows growth over time. What change would best enable the synthesis agent to correctly interpret such temporal differences?

D1 · Agentic Architecture & Orchestration

- **A** Require subagents to include publication or data collection dates in their structured outputs.
- **B** Add a conflict resolution agent that automatically discards older data when newer data exists for the same metric.
- **C** Configure the web search agent to only return results from the past 6 months.
- **D** Instruct the synthesis agent to always treat the most recent data as authoritative and place older findings in a separate historical appendix.

Correct answer: A

EXPLANATION

Why it's correct: A. Correct. The synthesis agent misreads the data because it never sees the dates. Making each data point carry its own timestamp in the structured output lets synthesis reason about trends instead of contradictions.

Option breakdown:

- **A:** Correct. The synthesis agent misreads the data because it never sees the dates. Making each data point carry its own timestamp in the structured output lets synthesis reason about trends instead of contradictions.
- **B:** Silently discarding older data destroys trend information — the very thing the question is asking about.
- **C:** Shrinking the window throws away historical context and doesn't fix the architectural gap that metadata isn't being passed to synthesis.
- **D:** Prompt instructions alone are unreliable and still hide temporal reasoning behind a rule. Structuring the data with dates is the systematic fix.

22 Production reviews reveal inconsistent handling of uncertainty in final reports. Sometimes conflicting subagent findings are synthesized into a single confident statement (losing nuance), while other times reports over-hedge with excessive qualifications (becoming unhelpful). When the web search agent returns "industry analysts estimate $50B market size (methodology varies)" and the document analysis agent returns "peer-reviewed study estimates 35B(±7B, 95% CI)," the coordinator either picks one arbitrarily or produces vague statements like "the market may be 35B–50B depending on factors." What systematic approach best addresses this?

D1 · Agentic Architecture & Orchestration

A Configure subagents to only report findings meeting a high-confidence threshold, filtering uncertain information before it reaches the coordinator.

B Implement a confidence calibration layer that normalizes subagent uncertainty expressions to standardized probability scores (0.0-1.0), then weight-average findings by their calibrated confidence.

C Instruct the synthesis agent to structure reports with explicit sections distinguishing well-established findings from contested ones, preserving original source characterizations and methodological context.

D Add a verification subagent that cross-references findings across sources, only passing claims to synthesis that are corroborated by at least two independent sources.

Correct answer: C

EXPLANATION

Why it's correct: C. Correct. Report structure that keeps methodological context and separates settled vs. contested claims is how you get nuance without over-hedging.

Option breakdown:

- **A:** Filtering throws away useful-but-uncertain evidence. It doesn't help the synthesis agent reason about conflicts — it just hides them.
- **B:** Collapsing peer-reviewed CIs and analyst estimates into a single averaged number destroys the methodological difference that's the whole point.
- **C:** Correct. Report structure that keeps methodological context and separates settled vs. contested claims is how you get nuance without over-hedging.
- **D:** Two-source minimums drop legitimate single-source findings and don't resolve genuinely conflicting estimates with different methodologies.

02

DOMAIN 2

Tool Design & MCP Integration

18% of scored content

20 questions

Tool design, built-in tools (Grep / Glob / Bash) and MCP servers, tools and resources.

23 You're building a security scanning workflow. When engineers need to locate all occurrences of a dangerous function like eval() across a large codebase, which tool should your agent use for content search?

D2 · Tool Design & MCP Integration

A. Use Glob with a pattern like /*eval* to find files, then Read each matching file.

B. Use Grep to search for the pattern "eval(" across all files in the codebase.

C. Read the project's main entry file and follow import statements to trace where eval might be used.

D. Use Bash to run ls -R | grep eval to recursively list files containing eval.

Correct answer: B

EXPLANATION

Grep is the purpose-built content-search tool (ripgrep-based) for finding patterns inside file contents across a large codebase, which is exactly what locating all eval(call sites requires. Glob only matches file names, reading entry files and tracing imports doesn't scale or guarantee coverage, and ls -R | grep only matches file names, not file contents.

24 This currently requires manually copy-pasting content into conversations. The team wants the agent to access this standard Jira ticket data directly. What's the most effective approach?

D2 · Tool Design & MCP Integration

Scenario: Developer Productivity — Engineers frequently ask the agent to cross-reference code changes with Jira tickets during reviews—checking ticket descriptions, acceptance criteria, and recent comments.

A. Export Jira tickets to markdown files in the repository that the agent accesses using the Read tool.

B. Build a custom MCP server wrapping Jira's API with tools designed specifically for this team's code review workflow.

C. Integrate an existing Jira MCP server that exposes tickets, comments, and metadata through discoverable tool interfaces.

D. Use the Bash tool with curl to call Jira's REST API, including authentication headers and parsing JSON responses inline.

Correct answer: C

EXPLANATION

For standard third-party service data like Jira tickets, Anthropic's recommended practice is to integrate an existing MCP server, which provides discoverable, typed tool interfaces with managed authentication. Building a custom MCP server (B) is unnecessary effort for standard ticket data, markdown exports (A) go stale and require manual syncing, and ad-hoc curl calls (D) are brittle and expose credentials inline.

25 Which built-in tool is most appropriate for this task?

D2 · Tool Design & MCP Integration

Scenario: Developer Productivity An engineer asks the agent to find all files in the monorepo that import the @company/auth package to understand how authentication is used across services.

A Bash, to execute find . -type d -name "*auth*" and explore matching directories

B Read, starting with package.json files to trace dependency declarations

C Grep, to search for the import statement pattern across file contents

D Glob, to find files with "auth" in their filename or path

Correct answer: C

EXPLANATION

The task is to find every file whose contents contain an import of @company/auth, which is a content- search problem — exactly what the built-in Grep tool (ripgrep-based) is designed for. Glob and find/Bash only match filenames or directory names, which would miss the many importing files that don't have "auth" in their path, and reading package.json files only shows declared dependencies, not actual usage sites.

26 Your productivity agent connects to three MCP servers: an issue tracker (search_issues, get_issue, create_comment), a documentation wiki (search_docs, get_page, list_spaces), and a database explorer (run_query, get_schema, list_databases). When engineers ask cross-system questions like "What database tables are affected by the authentication refactor in PROJ-1234?", monitoring shows the agent makes 8-10 sequential tool calls, frequently issues exploratory calls because it lacks visibility into what content each server contains, and exhausts context space before completing complex investigations. What architectural change best leverages MCP capabilities to address these issues?

D2 · Tool Design & MCP Integration

A. Consolidate all three servers into a unified MCP server with cross-referencing capabilities

B. Expose each server's content catalog as MCP resources—issue summaries, documentation hierarchy, database schemas

C. Add a prepare_investigation tool to each server that accepts a natural language question and returns relevant content summaries

D. Add an orchestrator that routes questions to a single server based on keywords

Correct answer: B

EXPLANATION

MCP resources are the protocol primitive designed for exactly this problem: servers expose browsable, application-controlled context (issue summaries, doc hierarchies, database schemas) that the agent can discover and read without exploratory tool-call round trips, directly fixing the visibility gap and context exhaustion. A rebuilds a monolith against MCP's composable design, C just adds more tools without leveraging MCP capabilities, and D's single-server keyword routing breaks inherently cross-system questions.

27 During testing, agents frequently call tools outside their specialization—the synthesis agent attempts web searches, and the report generator tries to analyze documents. What is the primary cause of this poor tool selection behavior?

D2 · Tool Design & MCP Integration

Scenario: Multi-Agent Research — You've configured the system so that all four subagents have access to the complete set of 18 tools.

A Choosing from 18 tools instead of 4-5 relevant ones increases decision complexity beyond reliable selection thresholds.

B The coordinator cannot track which capabilities each subagent has, leading to misrouted tasks.

C The agents' role descriptions in their system prompts conflict with having access to tools outside that role.

D The tool definitions consume too much context window space, leaving insufficient room for task content.

Correct answer: A

EXPLANATION

Anthropic's agent best practices recommend giving each subagent only a small, role-relevant tool set, because tool selection accuracy degrades as the number of available (especially irrelevant) tools grows. With all 18 tools exposed to every agent, the model faces a larger decision space and predictably picks tools outside its specialization — the root cause is selection complexity, not coordinator tracking, prompt conflicts, or context-window exhaustion.

28 A customer returns 4 hours after their initial session about the same billing dispute. The previous 32-turn session contains lookup_order results showing "Status: PENDING, Expected resolution: 24-48 hours." In testing, you observe that when resuming sessions with stale tool results, the agent often references the outdated data in responses (e.g., "I see your refund is still being processed") even after subsequent fresh tool calls return different information. What approach most reliably handles returning customers?

D2 · Tool Design & MCP Integration

A. Resume with full history but filter out previous tool_result messages before resuming, keeping only the human/assistant turns so the agent must re-fetch needed data.

B. Start a new session, inject a structured summary of the previous interaction (issue type, actions taken, resolution status), then make fresh tool calls before engaging.

C. Resume with full history and add a system prompt instruction telling the agent to always prefer the most recent tool results when multiple calls to the same tool exist in context.

D. Resume with full history and configure the agent to automatically re-call all previously-used tools at session start to ensure data freshness.

Correct answer: B

EXPLANATION

Why it's correct: B. Correct. A clean session with a summary keeps the narrative continuity while guaranteeing the agent isn't reasoning over stale tool results.

Option breakdown:

- **A:** Stripping tool_results from the middle of a conversation can leave assistant messages referencing nonexistent results — the transcript becomes internally inconsistent.
- **B:** Correct. A clean session with a summary keeps the narrative continuity while guaranteeing the agent isn't reasoning over stale tool results.
- **C:** Prompt instructions are suggestions. You observed that exact failure mode in testing — the model still references old results.
- **D:** Blanket re-calling is wasteful, slow, and still leaves the old results sitting in context to confuse the model.

29 You're implementing the escalation logic for when the agent should call escalate_to_human. Your team proposes four different approaches for triggering escalation. Which approach will most reliably identify cases that genuinely require human intervention?

D2 · Tool Design & MCP Integration

A Instruct the agent to escalate when the customer requests a human, when the issue requires policy exceptions, or when the agent cannot make meaningful progress.

B Configure the agent to escalate after three consecutive tool calls that fail to resolve the customer's stated issue, ensuring a reasonable attempt before involving a human.

C Implement sentiment analysis that monitors for frustration indicators (negative language, repeated questions, exclamation marks) and trigger escalation when the frustration score exceeds a configured threshold.

D Build a rules engine that maps specific issue types, customer segments, and product categories to escalation decisions, removing the need for model judgment calls.

Correct answer: A

EXPLANATION

Why it's correct: A. Correct. Escalation decisions are judgment calls about intent and progress — exactly what LLMs are good at. Clear criteria in natural language outperform rigid rules for the long tail.

Option breakdown:

- **A:** Correct. Escalation decisions are judgment calls about intent and progress — exactly what LLMs are good at. Clear criteria in natural language outperform rigid rules for the long tail.
- **B:** A hard retry count fires both too early (legitimate retries) and too late (obvious policy issues on the first call).
- **C:** Sentiment can catch frustration but misses calm customers who simply need a human for a policy exception, and over-escalates on stylistic language.
- **D:** Rules engines break on the cases they weren't designed for — and customer support is full of those.

30 After investigating a billing dispute over 25+ turns, you've identified that duplicate charges occurred due to a payment gateway timeout triggering retry logic. The required refund ($847) exceeds your $500 authorization limit. You need to call escalate_to_human, and the human agent won't have access to your conversation transcript. What context should you pass to enable effective resolution?

D2 · Tool Design & MCP Integration

A The customer's original complaint verbatim plus the tool result excerpts showing duplicate transactions.

B A structured summary: customer ID, root cause, refund amount, and recommended action.

C The complete conversation transcript with all tool results.

D Your diagnosis and the refund amount only.

Correct answer: B

EXPLANATION

Why it's correct: B. Correct. A structured handoff with identifiers, cause, amount, and recommended action is what a human agent needs to pick up the case instantly without re-investigating.

Option breakdown:

- **A:** Raw artifacts without synthesis force the human to re-do the 25 turns of investigation you just finished.
- **B:** Correct. A structured handoff with identifiers, cause, amount, and recommended action is what a human agent needs to pick up the case instantly without re-investigating.
- **C:** Dumping the whole transcript forces the human to wade through 25 turns instead of reading a one-screen brief.
- **D:** Missing customer identifiers and recommended action — the human can't act without them.

31 Compliance requires that refunds exceeding $500 must automatically escalate to a human agent—this rule cannot be left to model discretion. Despite clear system prompt instructions, production logs show the agent occasionally processes high-value refunds directly (3% failure rate). How should you achieve guaranteed compliance?

D2 · Tool Design & MCP Integration

A Modify the refund tool to return an error with message "Amount exceeds policy limit—please escalate" when threshold is exceeded.

B Add few-shot examples to the prompt showing correct escalation behavior at various refund amounts ($400, $500, $600).

C Implement a hook to intercept tool calls; when the refund process amount exceeds $500, block it and invoke human escalation.

D Strengthen the system prompt with emphatic language: "CRITICAL POLICY: Refunds over $500 MUST trigger human escalation. NEVER process these directly."

Correct answer: C

EXPLANATION

Why it's correct: C. Correct. Compliance-grade rules belong outside the model — a deterministic hook on the tool call is guaranteed to fire every time, independent of model behavior.

Option breakdown:

- **A:** This helps, but depends on the agent interpreting the error correctly and escalating — still model discretion at the decision point.
- **B:** Few-shots shift the distribution but don't remove the 3% failure. Compliance says the rule can't be left to model discretion.
- **C:** Correct. Compliance-grade rules belong outside the model — a deterministic hook on the tool call is guaranteed to fire every time, independent of model behavior.
- **D:** Emphatic prompts reduce but don't eliminate misuse. 'Cannot be left to model discretion' rules out any prompt-only solution.

32 During a billing dispute resolution, your agent successfully retrieves customer info via get_customer and order details via lookup_order, but when attempting to call process_refund, the tool returns a timeout error. The agent has enough information to explain the charges and verify refund eligibility, but cannot actually process the refund due to the backend failure. What approach best balances first-contact resolution with appropriate error handling?

D2 · Tool Design & MCP Integration

A. Escalate immediately to a human agent since the refund action cannot be completed

B. Implement automatic retries with exponential backoff for process_refund, keeping the conversation open until the refund is successfully processed

C. Explain the billing, confirm refund eligibility, acknowledge the system issue preventing immediate processing, and offer escalation or retry later

D. Confirm the refund will be processed and close the conversation, since the system has all necessary information to complete it automatically

Correct answer: C

EXPLANATION

Why it's correct: C. Correct. Deliver the partial value you can (explanation + eligibility), be honest about the failure, and let the customer choose between human escalation or a retry. Classic graceful degradation.

Option breakdown:

- **A:** Punts a transient backend issue to a human when the agent could still explain the situation and offer options.
- **B:** Forcing the customer to wait on indefinite retries is poor UX and may never succeed if the outage is long-lived.
- **C:** Correct. Deliver the partial value you can (explanation + eligibility), be honest about the failure, and let the customer choose between human escalation or a retry. Classic graceful degradation.
- **D:** Committing to an outcome that didn't happen is misleading the customer — a bigger failure than the timeout.

33 A customer writes: "I've been going back and forth on this return for days. I just want to speak to someone who can actually help me." The agent has confirmed via lookup_order that the return is straightforward—within policy and eligible for immediate processing. What should the agent do?

D2 · Tool Design & MCP Integration

- **A** Acknowledge frustration, inform them this is resolvable now, and offer to complete it or escalate
- **B** Call escalate_to_human immediately to honor the customer's request
- **C** Process the refund via process_refund to resolve the underlying issue, then inform them it's complete
- **D** Ask what specifically hasn't worked in previous attempts before deciding whether to escalate or resolve automatically

Correct answer: A

EXPLANATION

Why it's correct: A. Correct. Honor the feeling, give them the fast resolution path in writing, and preserve their choice. That's the customer-respect move that still leverages the agent's capability.

Option breakdown:

- **A:** Correct. Honor the feeling, give them the fast resolution path in writing, and preserve their choice. That's the customer-respect move that still leverages the agent's capability.
- **B:** Unnecessary queueing when the issue is one tool call away. Frustrates the customer further by adding waiting to an already-simple case.
- **C:** Takes unilateral action after the customer explicitly asked to speak to someone — overrides their stated preference.
- **D:** Interrogating a frustrated customer about past failures is the opposite of what they asked for.

34 The agent verifies customer identity through a multi-step process before resetting passwords. During testing, you notice that after the customer answers the third verification question, the agent asks them to provide their name again, as if the earlier exchange never happened. What's the most likely cause of this behavior?

D2 · Tool Design & MCP Integration

A. The verification tool is clearing the agent's internal state after each successful validation step.

B. The prompt lacks instructions telling Claude to remember information across multiple exchanges.

C. The conversation history isn't being passed in subsequent API requests.

D. Claude's memory retention is limited to two conversational turns by default, requiring explicit configuration to extend it.

Correct answer: C

EXPLANATION

Why it's correct: C. Correct. The API is stateless. Each request must include the full messages array. If you only send the latest turn, the model has no memory of earlier ones — exactly the 'ask for the name again' symptom.
Option breakdown:

- **A:** Tools don't clear agent state — conversation state lives in the messages array you send.
- **B:** Memory across turns isn't achieved by an instruction — it comes from passing the prior turns into the next request.
- **C:** Correct. The API is stateless. Each request must include the full messages array. If you only send the latest turn, the model has no memory of earlier ones — exactly the 'ask for the name again' symptom.
- **D:** No such default limit exists. Context is bounded by the window, and history length is under your control.

35 Production logs reveal inconsistent error handling: when lookup_order fails, the agent sometimes retries 5+ times (wasteful when the order ID doesn't exist), sometimes escalates immediately (premature for temporary network issues), and sometimes asks users for clarification (inappropriate when the issue is a backend permission error). Investigation shows your MCP tool returns uniform error responses: {"isError": true, "content": [{"type": "text", "text": "Operation failed"}]}. The agent cannot distinguish between error types. What's the most effective improvement?

D2 · Tool Design & MCP Integration

A Enhance error responses with structured metadata: include errorCategory (transient/validation/permission), isRetryable boolean, and a description of what caused the failure.

B Create an analyze_error MCP tool the agent calls after any failure to determine the error category and recommended action.

C Implement retry logic with exponential backoff in your MCP server for all errors, returning to the agent only after retries are exhausted.

D Add few-shot examples to the system prompt demonstrating how to interpret error message patterns and select appropriate responses for each.

Correct answer: A

EXPLANATION

Why it's correct: A. Correct. Give the agent the information it needs to make the right decision: category, retryability, and a human-readable cause. That replaces guessing with deterministic policy.

Option breakdown:

- **A:** Correct. Give the agent the information it needs to make the right decision: category, retryability, and a human-readable cause. That replaces guessing with deterministic policy.
- **B:** Adds an extra round-trip for something the original tool already knows. Put the metadata in the original response.
- **C:** Blanket retries hurt on permanent errors (order not found) and hide useful distinctions from the agent.
- **D:** If the tool returns 'Operation failed' for every failure, no amount of few-shots can extract category info that isn't there.

36 When the agent calls lookup_order and receives order details showing the item was purchased 45 days ago, how does the agentic loop determine whether to call process_refund or escalate_to_human next?

D2 · Tool Design & MCP Integration

A The orchestration layer automatically routes to the next tool based on the order's status field.

B The agent follows a pre-configured decision tree mapping order attributes to specific tool calls.

C The order details are added to the conversation and the model reasons about which action to take.

D The agent executes the remaining steps in a tool sequence planned at the start of the request.

Correct answer: C

EXPLANATION

Why it's correct: C. Correct. The agentic loop works by appending tool_result messages to the conversation and letting the model decide the next step on each turn. That's how 45 days refund vs. escalate gets resolved.

Option breakdown:

- **A:** There's no implicit orchestration layer picking tools from a field. The model drives tool selection.
- **B:** Agentic loops are model-driven, not decision-tree-driven. Hardcoded trees are the opposite of what the agent pattern is for.
- **C:** Correct. The agentic loop works by appending tool_result messages to the conversation and letting the model decide the next step on each turn. That's how 45 days refund vs. escalate gets resolved.
- **D:** There's no upfront, committed plan — the agent chooses each next step based on the latest context.

37 A customer sends: "This is frustrating. I've explained my issue twice and nothing is being resolved. I want to talk to a real person NOW." The agent has not yet called any tools to investigate their account. What should the agent do?

D2 · Tool Design & MCP Integration

- **A** Acknowledge the frustration and ask one targeted question to understand the specific issue before escalating.
- **B** Briefly explain what the agent can help with and offer to resolve the issue quickly, escalating only if the customer repeats their request.
- **C** Immediately call escalate_to_human with the conversation history.
- **D** First call get_customer and lookup_order to gather account context, then escalate to a human agent.

Correct answer: A

EXPLANATION

Why it's correct: A. Correct. The customer has said 'twice' but you have no context yet. One acknowledging, focused question gives you a shot at first-contact resolution without dismissing the frustration or delaying a potential handoff.

Option breakdown:

- **A:** Correct. The customer has said 'twice' but you have no context yet. One acknowledging, focused question gives you a shot at first-contact resolution without dismissing the frustration or delaying a potential handoff.
- **B:** Launching into capability-listing at a frustrated customer who asked for a human reads as dismissive.
- **C:** Escalating with zero tool context creates a cold handoff where the human also starts from scratch.
- **D:** Investigating without asking adds latency and doesn't respect the customer's request. A single acknowledging question is faster and more respectful.

38 Your agent is handling a billing dispute. After calling get_customer and lookup_order, it identifies that the dispute involves a promotional pricing error requiring manager approval—beyond the agent's authorization level. How should the workflow handle this mid-process escalation?

D2 · Tool Design & MCP Integration

A. Call escalate_to_human passing only the customer's original message.

B. Compile a structured handoff with customer details, order info, and the identified issue before calling escalate_to_human.

C. Attempt the refund with process_refund anyway, escalating only if the system rejects the transaction.

D. Persist the complete conversation and tool response history to a database, then call escalate_to_human with a reference ID.

Correct answer: B

EXPLANATION

Why it's correct: B. Correct. A structured brief (who, what order, what issue, why it exceeds auth) lets the human agent pick up instantly. That's the mid-process escalation pattern.

Option breakdown:

- **A:** Drops the tool-derived context the agent just gathered, forcing the human to re-do the investigation.
- **B:** Correct. A structured brief (who, what order, what issue, why it exceeds auth) lets the human agent pick up instantly. That's the mid-process escalation pattern.
- **C:** Knowingly exceeding authorization is a policy violation — not something to try and hope the system catches.
- **D:** Adds infrastructure and an extra lookup step when an inline structured brief is simpler and faster.

39 A customer raises three separate issues during one session: a refund inquiry (turns 1-15), a subscription question (turns 16-30), and a payment method update (turns 31-45). At turn 48, the customer asks "What happened with my refund?" The conversation is approaching context limits. What strategy best maintains the agent's ability to address all issues throughout the session?

D2 · Tool Design & MCP Integration

A. Extract and persist structured issue data (order IDs, amounts, statuses) into a separate context layer.

B. Rely on MCP tools to re-fetch relevant information on demand when the customer references earlier issues.

C. Summarize earlier turns into a narrative description, preserving full message history only for the active issue.

D. Implement sliding window context that retains the most recent 30 turns.

Correct answer: C

EXPLANATION

Why it's correct: C. Correct. Progressive summarization compresses stable resolved topics while keeping the active thread verbatim — the classic pattern for long multi-issue conversations near the context limit.

Option breakdown:

- **A:** A sidecar context layer adds engineering and doesn't naturally preserve the conversational narrative the customer expects.
- **B:** Re-fetching is fine for freshness, but it doesn't solve the context-length problem — and you lose what was said between the customer and the agent.
- **C:** Correct. Progressive summarization compresses stable resolved topics while keeping the active thread verbatim — the classic pattern for long multi-issue conversations near the context limit.
- **D:** A pure sliding window silently drops the refund issue (turns 1–15) from context, which is exactly what the customer is asking about.

40 When implementing your lookup_order MCP tool, the backend sometimes returns errors (e.g., "Order not found" or temporary database failures). What is the correct pattern for communicating these errors back to the agent?

D2 · Tool Design & MCP Integration

A Log the error server-side and return an empty result to avoid confusing the model

B Return the error message in the tool result content with the isError flag set to true

C Throw an exception from the tool handler so the agent framework can catch and log it

D Return a success response with a "status" field indicating the error type

Correct answer: B

EXPLANATION

Why it's correct: B. Correct. MCP's designed pattern: put the error text in the content field and mark isError=true. Claude sees both the failure flag and a readable message to reason about.

Option breakdown:

- **A:** Returning empty successes makes the agent think no data exists, rather than that something went wrong — a different and worse confusion.
- **B:** Correct. MCP's designed pattern: put the error text in the content field and mark isError=true. Claude sees both the failure flag and a readable message to reason about.
- **C:** Uncaught exceptions break the tool protocol and don't give the model anything to reason with.
- **D:** Ad-hoc 'status' fields vary across tools and the model has no standard way to interpret them. isError is the standard.

41 Your process_refund tool returns two types of errors: technical errors ("503 Service Unavailable", "Connection timeout") that are transient (5% of calls), and business errors ("Order exceeds 30-day return window", "Item already refunded") that are permanent (12% of calls). Monitoring shows the agent wastes 3-4 turns retrying business errors that can never succeed. Currently, both error types return only a plain text message to Claude. What's the most effective way to reduce wasted retries while improving customer-facing response quality?

D2 · Tool Design & MCP Integration

A Return structured error responses with retryable: false for business errors and a customer-friendly explanation for Claude to use.

B Add few-shot examples showing how to distinguish retryable from non-retryable errors by parsing error message text.

C Add a check_refund_eligibility tool that must be called before process_refund to prevent business rule violations.

D Implement automatic retry logic at the tool level for technical errors only, passing business errors to Claude without retries.

Correct answer: A

EXPLANATION

Why it's correct: A. Correct. A retryable flag tells Claude deterministically 'don't retry,' and a ready-made customer-friendly message improves the outgoing reply. Fixes both problems at once.

Option breakdown:

- **A:** Correct. A retryable flag tells Claude deterministically 'don't retry,' and a ready-made customer-friendly message improves the outgoing reply. Fixes both problems at once.
- **B:** Relying on the model to text-parse error strings is fragile and exactly the instability you see today.
- **C:** Useful in principle but adds a round-trip to every refund to guard against 12% of cases, and doesn't help when process_refund still fails for other business reasons.
- **D:** Hiding transient retries inside the tool can mask latency and takes the model out of the loop on recovery decisions. Business errors also still arrive as a plain string, so the customer-facing response doesn't improve.

42 Your agent has called lookup_order multiple times while investigating a customer's return requests. Each response includes 40+ fields (items, shipping details, payment info, status history). Tool outputs now represent the majority of the conversation's context. The customer mentions two more orders they want to discuss. What's the most effective approach before making additional lookups?

D2 · Tool Design & MCP Integration

- **A** Extract only return-relevant fields (items, purchase date, return window, status) from each existing order response, removing verbose details
- **B** Have the model generate a natural language summary of each order's key details, replacing structured responses with prose descriptions
- **C** Move all tool responses to a vector database with semantic indexing, retrieving relevant portions as the conversation continues
- **D** Proceed with additional lookups without modifying the existing tool output context

Correct answer: A

EXPLANATION

Why it's correct: A. Correct. Keep the fields that matter for the task and drop the rest. This directly addresses the context-bloat problem before you add two more lookups.

Option breakdown:

- **A:** Correct. Keep the fields that matter for the task and drop the rest. This directly addresses the context-bloat problem before you add two more lookups.
- **B:** Prose summaries lose precision for fields the model may need later (exact dates, amounts). Structured pruning is better than paraphrasing.
- **C:** A vector DB is heavy infrastructure for what is essentially a pruning problem.
- **D:** Does nothing about the bloat — you're heading for context exhaustion.

03

DOMAIN 3

Claude Code Configuration & Workflows

20% of scored content

18 questions

Claude Code flags, CLAUDE.md, sessions, permissions and developer workflows.

43 During initial testing of the automated review pipeline, you notice that reviews on large PRs (50+ changed files) sometimes take over 20 minutes and cost $8-12 per run due to extensive agentic loops — Claude reads files, runs analysis tools, and iterates many times. Your team needs each invocation to abort once it reaches a fixed iteration count and a fixed dollar amount, enforced by Claude Code itself rather than the surrounding job runner. Which configuration change directly enforces both of those per-invocation caps?

D3 · Claude Code Configuration & Workflows

- **A** Add --max-turns 10 --max-budget-usd 2.00 to the claude -p invocation to cap iterations and spend.
- **B** Switch the --model flag to a smaller, cheaper model so each iteration uses fewer tokens and lower per- call cost.
- **C** Set timeout-minutes: 5 on the GitHub Actions job step and monitor per-run costs via the Anthropic Console usage dashboard.
- **D** Set --permission-mode dontAsk to auto-deny any tool permission requests not in the explicitly allowed set.

Correct answer: A

EXPLANATION

Claude Code's --max-turns flag caps the number of agentic iterations and --max-budget-usd caps per- invocation dollar spend, both enforced natively by Claude Code in headless (claude -p) mode — exactly the two per-invocation limits required. Option C relies on the surrounding job runner (GitHub Actions timeout) and passive dashboard monitoring, which the question explicitly excludes, while B only lowers cost without enforcing any cap and D governs tool permissions, not iterations or spend.

44 Your test generation produces unit tests for new code, but reviews show 55% are low-value: trivial assertions that only verify functions don't throw exceptions, tests duplicating existing coverage, or tests ignoring your team's fixture conventions. How do you reduce the rate of low-value tests being generated in the first place?

D3 · Claude Code Configuration & Workflows

A. Add post-generation coverage analysis that automatically filters out any generated test that doesn't increase line coverage beyond what existing tests provide.

B. Restrict test generation to directories where historical quality metrics show higher acceptance rates, disabling it for areas where generated tests consistently require heavy editing.

C. Document testing standards in CLAUDE.md including valuable test criteria, available fixtures with intended use cases, and examples distinguishing meaningful behavioral tests from trivial assertions.

D. Implement a two-phase generation where a second Claude call scores each test against quality criteria, filtering out low-scoring tests before presenting results to developers.

Correct answer: C

EXPLANATION

The question explicitly asks how to reduce low-value tests "being generated in the first place," which means fixing the upstream cause: Claude lacks context about what the team considers a valuable test. Documenting testing standards, fixture usage, and good-vs-trivial examples in CLAUDE.md gives Claude that context at generation time, whereas A and D only filter bad output after the fact and B avoids the problem rather than solving it.

45 An engineer used Claude Code yesterday to investigate authentication flows in a legacy monolith, building up significant context over a 2-hour session. Today she wants to continue that specific investigation. She's worked on three other codebases since then and knows the session was named "auth-deep-dive". How should she resume?

D3 · Claude Code Configuration & Workflows

A. Use --continue to pick up where the most recent conversation left off

B. Use --resume auth-deep-dive to load that specific session by name

C. Use --session-id with the UUID from yesterday's session transcript file

D. Start fresh and re-read the same files

Correct answer: B

EXPLANATION

Claude Code's --resume flag accepts a session name or ID, so --resume auth-deep-dive loads that exact named session with all its accumulated context. --continue only resumes the most recent conversation, which would be one of the other sessions she's worked on since, and manually hunting for a UUID or starting fresh discards the value of the named session.

46 After integrating a local MCP server providing code analysis tools (analyze_dependencies, find_dead_code, calculate_complexity), you verify the server is healthy and tools appear in the tools/list response. However, you observe that the agent consistently uses Grep to search for import statements instead of calling analyze_dependencies—even when users explicitly ask about "code dependencies." Examining tool definitions reveals: - MCP: analyze_dependencies — "Analyzes dependency graph" - Built-in: Grep — "Search file contents for a pattern using regular expressions. Returns matching lines with line numbers and surrounding context." What's the most effective approach to improve the agent's selection of MCP tools?

D3 · Claude Code Configuration & Workflows

A Remove Grep from available tools when the MCP server is connected to eliminate functional overlap.

B Split analyze_dependencies into granular tools (list_imports, resolve_transitive_deps, detect_circular_deps) so each has a focused purpose less likely to overlap with Grep.

C Add routing instructions to the system prompt specifying that dependency-related questions should use MCP tools rather than Grep.

D Expand MCP tool descriptions to detail capabilities and outputs—e.g., "Builds dependency graph showing direct imports, transitive dependencies, and cycles."

Correct answer: D

EXPLANATION

The root cause is the asymmetry in tool descriptions: the MCP tool's vague "Analyzes dependency graph" loses to Grep's detailed, concrete description, so the agent picks Grep. Anthropic's tool-use best practices say the tool description is the primary lever for tool selection—writing detailed descriptions of capabilities, outputs, and when to use the tool (option D) fixes selection at the source, whereas removing Grep, fragmenting the tool, or adding brittle system-prompt routing rules are workarounds.

47 An engineer asks the agent to find all callers of a function before removing it. The function is defined in a core library but is also exposed through wrapper modules that rename the function for domain-specific use (e.g., calculateTax in the library becomes computeOrderTax in the orders module). What exploration strategy will most reliably identify all callers?

D3 · Claude Code Configuration & Workflows

A. Search for the function name in project documentation to understand intended usage patterns and navigate to documented integration points.

B. Use Grep to search for the function's original name across the codebase.

C. Read the library and wrapper modules to identify all exposed names for the function, then Grep for each name across the codebase.

D. Use Grep to find all files that import from the library or wrapper modules, then read each file to check whether it uses the function.

Correct answer: C

EXPLANATION

Because the function is re-exported under different names by wrapper modules, grepping only the original name (B) misses aliased callers, and documentation (A) may be stale or incomplete. The reliable strategy is to first read the library and wrappers to enumerate every exposed name/alias, then Grep for each name across the codebase (C); option D is indirect and far more laborious, requiring a manual read of every importing file.

48 An engineer's exploration subagent spent 30 minutes analyzing a legacy payment system, reading 47 files and documenting data flows. The session was interrupted when the engineer's connection dropped. While away, a teammate merged a PR that renamed two utility functions. The engineer wants to continue the same exploration. What's the most effective approach?

D3 · Claude Code Configuration & Workflows

- **A** Resume the subagent from its previous transcript and inform it about the renamed functions. ĐÁP ÁN ĐÚNG
- **B** Launch a fresh subagent and include the prior transcript in the initial prompt for context.
- **C** Launch a fresh subagent with a summary of prior findings.
- **D** Resume the subagent from its previous transcript without mentioning the changes—the architecture understanding remains valid.

Correct answer: A

EXPLANATION

Resuming from the saved transcript preserves all 30 minutes of accumulated exploration context (47 files, documented data flows) without re-spending tokens, and explicitly telling the agent about the renamed utility functions corrects the only part of its knowledge that became stale. D leaves the agent operating on outdated function names, B wastefully stuffs an entire transcript into a fresh context window, and C discards detailed file-level understanding.

49 An engineer now asks it to understand how the physics engine integrates with rendering for collision debug overlays. You notice recent responses reference "typical rendering patterns" rather than the specific VulkanPipeline and FrameGraph classes it discovered earlier. What's the most effective approach?

D3 · Claude Code Configuration & Workflows

Scenario: Developer Productivity — Your agent has spent 25 minutes exploring a game engine's rendering subsystem—reading shader code, buffer management, and frame synchronization logic.

A. Use /clear to reset context completely, then start fresh with physics exploration using file paths from the project's CLAUDE.md.

B. Spawn a sub-agent to explore physics independently, then manually synthesize its findings with the rendering knowledge accumulated in the main conversation.

C. Summarize key rendering findings, then spawn a sub-agent for physics exploration with that summary in its initial context.

D. Continue in the current context with more targeted prompts referencing the specific classes by name.

Correct answer: C

EXPLANATION

The generic "typical rendering patterns" responses signal context degradation from a long, bloated session, so the fix is to compact the valuable rendering knowledge into a summary and hand it to a sub-agent with a fresh context for the physics task. This follows Anthropic's context-management best practice of summarization plus sub-agents with clean contexts, preserving the specific VulkanPipeline/FrameGraph findings that the integration task depends on, unlike A (loses knowledge), B (sub-agent lacks needed rendering context and synthesis happens in the degraded context), or D (continues in a degraded context).

50 An engineer asks the agent to understand how the caching layer works before adding a new cache invalidation trigger. After initial Grep searches, the agent has identified that caching logic spans 15 files including decorators, middleware, and service classes (~8,000 lines total). What's the most effective next step for building understanding while managing context constraints?

D3 · Claude Code Configuration & Workflows

A. Use Glob to find files matching common caching patterns (cache.py, caching/), prioritize the largest files by reading them first, then check smaller files for gaps.

B. Use the Read tool to sequentially load all 15 files, building complete understanding across the full caching implementation.

C. Use Grep to search for "invalidate" and "expire" patterns across all files, then Read only those specific line ranges with minimal surrounding context.

D. Analyze imports and class hierarchies to identify the base cache class, Read that file to understand the interface, then trace specific invalidation implementations.

Correct answer: D

EXPLANATION

Starting from the base cache class gives the agent the core abstraction and interface that all 15 files build on, then tracing only the relevant invalidation implementations yields deep structural understanding with minimal context usage. Reading all files (B) wastes context, size-based prioritization (A) is arbitrary, and keyword-only line- range reads (C) produce fragmentary knowledge inadequate for safely adding a new invalidation trigger.

51 Your agent has analyzed a complex service module—reading 23 source files, tracing request flows, and identifying error handling patterns. A developer wants to compare two testing strategies before committing to one: end-to-end tests with mocked external services vs. snapshot tests capturing expected outputs. They need to independently develop both approaches to evaluate trade-offs. How should you manage the sessions?

D3 · Claude Code Configuration & Workflows

A. Export the analysis session's key findings to a file, then create two new sessions that reference this file.

B. Resume the analysis session with fork_session enabled, creating a separate branch for each testing strategy.

C. Start two fresh sessions, having each re-read the relevant source files before beginning.

D. Continue in the original session, developing end-to-end tests first, then snapshot tests sequentially.

Correct answer: B

EXPLANATION

Resuming the analysis session with fork_session (CLI --fork-session / Agent SDK fork_session=True with resume) branches the session into new session IDs while preserving the full analysis context and leaving the original intact. Forking twice gives each testing strategy the complete 23-file analysis with no re-work, and the two branches develop completely independently — exactly the stated requirement. Exporting to a file is lossy, fresh sessions discard the analysis, and a single sequential session is neither independent nor unbiased.

52 Your codebase exploration tool stores session IDs to allow engineers to continue investigations across work sessions. An engineer spent an hour yesterday analyzing a legacy authentication module, building context about its architecture and dependencies. They want to continue today. The session ID is valid, but version control shows 3 of the 12 files the agent previously read were modified overnight by a teammate's merge. What approach best balances efficiency and accuracy?

D3 · Claude Code Configuration & Workflows

Scenario: Developer Productivity.

A Resume the session and immediately have the agent re-read all 12 previously analyzed files

B Start a fresh session to ensure the agent works with current codebase state without stale assumptions

C Resume the session and inform the agent which specific files changed for targeted re-analysis
ĐÁP ÁN ĐÚNG

D Resume the session without informing the agent about the changed files

Correct answer: C

EXPLANATION

Resuming the session preserves the hour of accumulated context about the module's architecture (efficiency), while telling the agent exactly which 3 files changed allows targeted re-reading of only the stale files (accuracy). Re-reading all 12 files (A) wastes tokens, starting fresh (B) throws away valuable context, and resuming silently (D) risks the agent acting on stale file contents.

53 During testing, you observe that in extended exploration sessions (30+ minutes), the agent starts giving inconsistent answers about code structure it discussed earlier. Engineers report having to repeat context about modules they've already explored. What's the most effective approach to address this?

D3 · Claude Code Configuration & Workflows

- **A** Have the agent maintain a scratchpad file that records key findings, referencing it for subsequent questions.
- **B** Switch to a higher-capacity model tier to provide more context window space for accumulated exploration data.
- **C** Implement automatic context clearing every 15 minutes to ensure the agent starts with fresh, uncontaminated context.
- **D** Create summaries of all source files before exploration begins, loading only these compressed representations into context.

Correct answer: A

EXPLANATION

Why it's correct: A. Correct. A scratchpad offloads findings to durable storage the agent can re-read on demand, giving it a stable 'memory' independent of how crowded the context window gets.

Option breakdown:

- **A:** Correct. A scratchpad offloads findings to durable storage the agent can re-read on demand, giving it a stable 'memory' independent of how crowded the context window gets.
- **B:** A bigger window delays the problem but doesn't solve attention degradation over long, noisy contexts.
- **C:** Wholesale clearing throws away valid findings — exactly the state engineers complain about having to re-establish.
- **D:** Pre-summaries lose the detail that makes code exploration useful and often misrepresent files the engineer cares about most.

54 An engineer used the agent yesterday to analyze a legacy authentication module, identifying two distinct refactoring approaches: extracting a microservice versus refactoring in-place. Today, they want to explore both approaches in depth—having the agent propose specific code changes for each—before deciding which to implement. What's the most effective way to structure this exploration?

D3 · Claude Code Configuration & Workflows

A Resume yesterday's session to explore the first approach, then start a new session for the second, manually recreating the original context.

B Start two fresh sessions, manually providing a summary of yesterday's analysis findings to establish context.

C Resume yesterday's session and explore both approaches sequentially within the same conversation thread.

D Use fork_session to create two branches from yesterday's analysis, exploring one approach in each fork.

Correct answer: D

EXPLANATION

Why it's correct: D. Correct. Forking from yesterday's session gives each approach its own independent context starting from the same analysis baseline — clean, parallel, no contamination.

Option breakdown:

- **A:** Manual recreation is error-prone and loses the exact working state of yesterday's analysis.
- **B:** Redoes work and risks the two sessions diverging from the same baseline you established yesterday.
- **C:** Sequential exploration in one thread lets each approach contaminate the other's context.
- **D:** Correct. Forking from yesterday's session gives each approach its own independent context starting from the same analysis baseline — clean, parallel, no contamination.

55 An engineer asks your agent to identify untested code paths in a legacy payment processing module spanning 45 files. After reading the first 8 source files, the agent's responses are becoming noticeably less accurate—it's forgetting previously discussed code patterns and hasn't yet located all test files or traced critical payment flows. What's the most effective approach to complete this investigation?

D3 · Claude Code Configuration & Workflows

A. Document all current findings in a summary report, clear context completely, then use that report as the sole reference for continuing the investigation.

B. Spawn subagents to investigate specific questions (e.g., "find all test files for payment processing", "trace refund flow dependencies") while the main agent coordinates findings and preserves high-level understanding.

C. Clear context with /clear, then selectively re-read only the most critical files discovered so far, writing key findings to a scratchpad file that persists between context resets.

D. Switch to using Grep to search for specific function names instead of reading full files, reducing the content loaded into context for remaining exploration.

Correct answer: B

EXPLANATION

Why it's correct: B. Correct. Delegate well-scoped investigations to subagents with fresh context, while the main agent keeps the architectural overview. This is the pattern for scaling exploration beyond a single context window.

Option breakdown:

- **A:** A single report becomes the only source of truth and tends to compress away the specific code patterns you'd need later.
- **B:** Correct. Delegate well-scoped investigations to subagents with fresh context, while the main agent keeps the architectural overview. This is the pattern for scaling exploration beyond a single context window.
- **C:** Scratchpads help, but clearing + re-reading discards the understanding you already built across 8 files.
- **D:** You can't identify untested paths just by grepping names — you need to read enough of each path to know what branches the tests don't cover.

56 A developer asks the agent to investigate why a specific API endpoint intermittently returns 500 errors. The codebase has 200+ files and the developer doesn't know which components are involved. The agent must trace the error through routing, middleware, business logic, and database layers. What task decomposition approach would be most effective?

D3 · Claude Code Configuration & Workflows

A. Have the agent first create a comprehensive plan mapping all code paths through the endpoint before beginning any file exploration or code reading.

B. Have the agent dynamically generate investigation subtasks based on what it discovers at each step, adapting its exploration plan as new information about the error path emerges.

C. Define a fixed sequence of investigation steps upfront—grep for error patterns, then read error handlers, then check database queries, then examine middleware—executing each step regardless of intermediate findings.

D. Run parallel worker agents that simultaneously investigate all four layers, then synthesize their findings to identify where the error originates.

Correct answer: B

EXPLANATION

Why it's correct: B. Correct. Debugging is adaptive by nature — each file you read changes the most useful next step. Let the agent follow the evidence.

Option breakdown:

- **A:** You can't build a correct plan for an unknown error without any exploration. Planning blind wastes time and misses the actual failure path.
- **B:** Correct. Debugging is adaptive by nature — each file you read changes the most useful next step. Let the agent follow the evidence.
- **C:** A fixed pipeline wastes work on layers that aren't involved and can lock the agent out of the actual root cause path.
- **D:** Fan-out is useful once you have well-scoped subtasks. Here, you don't — you'd pay 4× the cost to look in three layers that aren't the problem.

57 After adding an MCP server with specialized code refactoring tools (extract_function, rename_variable, inline_function), you notice the agent still uses basic text manipulation via Write and Bash sed commands for refactoring tasks. The MCP server is connected and healthy. Examining the configuration, you find each MCP tool has a minimal description like "extract_function: extracts a function from code." What's the most effective way to improve adoption of the MCP refactoring tools?

D3 · Claude Code Configuration & Workflows

A. Implement a request classifier that detects refactoring intent and automatically routes those requests to the MCP server before the agent processes them.

B. Remove the Write tool from the agent's configuration for refactoring sessions so it must use the MCP tools for code modifications.

C. Accept this as expected behavior since simpler tools like sed are more predictable than specialized refactoring tools.

D. Enhance the MCP tool descriptions to explain when each tool is preferable to text manipulation and clarify expected inputs and outputs.

Correct answer: D

EXPLANATION

Why it's correct: D. Correct. Tool selection is driven by the descriptions Claude sees. When the MCP tools say 'extracts a function from code' and Write/sed come with rich documentation, Claude picks Write/sed. Beef up the descriptions.

Option breakdown:

- **A:** A pre-classifier is a separate system to maintain and can misroute. The cheaper fix is to make the tool descriptions strong enough that the agent picks them on its own.
- **B:** Stripping general tools forces adoption by subtraction, and breaks legitimate Write use cases.
- **C:** Capitulating defeats the purpose of integrating refactoring tools. The integration is fine — the descriptions are the problem.
- **D:** Correct. Tool selection is driven by the descriptions Claude sees. When the MCP tools say 'extracts a function from code' and Write/sed come with rich documentation, Claude picks Write/sed. Beef up the descriptions.

58 Your agent needs to insert a new helper function into the middle of a 150-line utility module, between two existing functions. The Edit tool fails because its old_string parameter cannot find unique text to match — the file has repetitive docstrings, variable names, and structural patterns. What's the most reliable way to complete this insertion?

D3 · Claude Code Configuration & Workflows

A Use Edit with an extremely long old_string capturing 30+ lines of context to guarantee uniqueness

B Use Edit's replace_all parameter to target a common pattern and embed the new function in the replacement text

C Use Bash to append the function definition to the end of the file using heredoc syntax

D Use Read to load the file, add the function at the appropriate location, then Write the updated file

Correct answer: D

EXPLANATION

Why it's correct: D. Correct. When Edit's unique-match contract can't be satisfied in a repetitive file, fall back to Read modify in memory at the intended line Write the full file back.

Option breakdown:

- **A:** Long, brittle match strings frequently miss due to whitespace or minor edits and produce confusing failures.
- **B:** replace_all would mutate every occurrence of the pattern — corrupting the whole file.
- **C:** Appending puts the function at the wrong location. The requirement is to insert between two existing functions.
- **D:** Correct. When Edit's unique-match contract can't be satisfied in a repetitive file, fall back to Read modify in memory at the intended line Write the full file back.

59 An engineer who just joined the team asks the agent to help them understand the authentication and authorization architecture before making security improvements. The codebase has 800+ files across multiple services. What exploration strategy will most effectively build understanding, given Claude built-in tools and context limits?

D3 · Claude Code Configuration & Workflows

A. Read any CLAUDE.md and README files first, then ask the engineer to specify which 10-15 files are most important for understanding the auth system.

B. Launch parallel subagents to explore different services simultaneously, then synthesize their findings into an architectural overview.

C. Use Grep to find authentication entry points, read those files, then follow imports and function calls to map the auth flow incrementally.

D. Read all files containing "auth", "login", "permission", or "token" in their content or filename.

Correct answer: C

EXPLANATION

Why it's correct: C. Correct. Start at entry points (login, token verify, middleware), then trace outward following real code edges. Incremental, grounded, fits within context limits.

Option breakdown:

- **A:** The engineer just joined — they probably don't know which files matter. That's exactly what the agent is supposed to help with.
- **B:** Without knowing where auth lives, fan-out explores too broadly and subagents duplicate and miss cross-service flows.
- **C:** Correct. Start at entry points (login, token verify, middleware), then trace outward following real code edges. Incremental, grounded, fits within context limits.
- **D:** Those keywords hit a huge amount of unrelated code across 800+ files and drown context in noise.

60 A critical bug is affecting production users. Error logs show exceptions in the OrderProcessing module with a clear stack trace pointing to a specific area, but you haven't worked with this module before. What's the most effective approach?

D3 · Claude Code Configuration & Workflows

A. Use plan mode to analyze the error in context of the module's design, enumerate potential root causes, and prioritize fixes systematically.

B. Start with direct execution to gather initial information, then switch to plan mode to design a comprehensive solution before implementing.

C. Use direct execution to examine the stack trace, read the relevant code, and implement a fix once you identify the root cause.

D. Enter plan mode to explore the module's architecture and dependencies before attempting any fix.

Correct answer: C

EXPLANATION

Why it's correct: C. Debugging is adaptive — each file you read changes the most useful next step, so let the agent follow the evidence. The stack trace already points straight at the failure, so direct execution (read the trace read the relevant code fix once the root cause is identified) is the fastest, most reliable path.

Option breakdown:

- **A:** Planning before any exploration of an unknown error is planning blind; it wastes time and easily misses the real failure path.
- **B:** Adding a plan-mode step is unnecessary for an already-localized bug and slows down incident response on a production outage.
- **C:** Correct — direct execution follows the evidence and fixes the bug as soon as the root cause is found.
- **D:** Exploring the whole architecture and dependencies is over-exploration when the stack trace has already pinpointed the location.

04

DOMAIN 4

Prompt Engineering & Structured Output

20% of scored content

14 questions

Prompt engineering, few-shot examples, structured output, confidence and severity metadata.

61 After deploying the automated review, you notice high precision but low recall — real bugs are slipping through undetected. Investigation reveals your review prompt instructs Claude to "only report high- confidence issues you are certain about" and "err on the side of not commenting." Developers appreciate the low noise, but a race condition that caused a production outage was visible in a reviewed PR and went unreported. You need to substantially improve bug detection while keeping false positive rates manageable for your team. What is the most effective approach?

D4 · Prompt Engineering & Structured Output

A. Remove the conservative filtering instructions and prompt Claude to report all potential issues, then apply a programmatic filter to deduplicate and suppress categories that historically generate false positives.

B. Add detailed few-shot examples demonstrating bug categories Claude should flag — race conditions, null dereferences, error handling gaps — while keeping the high-confidence filtering instruction to maintain current precision levels.

C. Expand the context window by including related test files, recent git history, and the module's dependency graph alongside the diff, giving Claude richer signals to assess issue severity.

D. Split the review into a finding stage where Claude's goal is coverage — flagging every potential issue with confidence and severity metadata — and a separate stage that thresholds those findings. ĐÁP ÁN ĐÚNG

Correct answer: D

EXPLANATION

Anthropic's code-review guidance for recent Claude models states that "only report high-confidence" instructions are followed literally and depress recall, and recommends telling Claude to report every finding with confidence and severity metadata while moving filtering to a separate downstream verification/thresholding stage. Option D is exactly this two-stage coverage-then-threshold pattern; B keeps the recall-killing filter, C doesn't address the filtering instruction, and A's blunt category suppression can re-suppress real bugs and lacks the confidence/severity metadata needed for principled filtering.

62

Your extraction system processes two document types: standard monthly reports (archived after processing) and urgent exception reports (must trigger business alerts within 30 minutes of receipt). Both use the same JSON schema. You want to minimize API costs while meeting latency requirements. How should you architect the processing pipeline?

D4 · Prompt Engineering & Structured Output

A. Submit all documents to the real-time Messages API to ensure consistent processing latency across document types.

B. Submit all documents to the Batch API with custom_ids for tracking. When results arrive, immediately process urgent documents and trigger delayed alerts for exceptions.

C. Queue all documents and submit hourly batches, flagging urgent documents for expedited handling when batch results return.

D. Route standard reports to the Batch API for 50% cost savings, and route urgent exception reports to the real-time Messages API.

Correct answer: D

EXPLANATION

Why it's correct: D. Correct. Match latency profile to document urgency: batch for the bulk (cheap), real-time for the latency-sensitive exceptions (fast). Minimizes cost while meeting SLA.

Option breakdown:

- **A:** Consistent but expensive. Standard monthly reports don't need real-time latency and shouldn't pay for it.
- **B:** Batch has up to a 24-hour window — 30-minute alerting SLAs on exception reports can't be met by routing them through batch.
- **C:** Hourly batches still ride the batch SLO and don't guarantee the 30-minute alert window for exceptions.
- **D:** Correct. Match latency profile to document urgency: batch for the bulk (cheap), real-time for the latency-sensitive exceptions (fast). Minimizes cost while meeting SLA.

63 Your schema includes a skills: string[] field. Production monitoring reveals three consistency issues: (1) compound phrases like "Python and SQL" are sometimes kept as one entry, sometimes split; (2) implied but unstated skills occasionally appear in extractions; (3) similar documents produce wildly different array lengths (5-10 vs 40+ entries). Your prompt currently says "Extract all skills mentioned." What's the most effective improvement?

D4 · Prompt Engineering & Structured Output

- **A** Add few-shot examples demonstrating compound phrase handling, explicit mention criteria, and appropriate entry granularity.
- **B** Add constraints: "Extract 10-20 skills maximum, one skill per entry, only explicitly named skills."
- **C** Add post-extraction normalization that maps skills to a canonical taxonomy and deduplicates similar entries.
- **D** Enrich the schema to {skill: string, confidence: float, source_quote: string}[] to capture extraction metadata.

Correct answer: A

EXPLANATION

Why it's correct: A. Correct. All three issues are about the model's interpretation of what counts as 'a skill.' Few-shot examples teach the pattern concretely — split vs. not-split, mentioned vs. inferred, appropriate granularity.

Option breakdown:

- **A:** Correct. All three issues are about the model's interpretation of what counts as 'a skill.' Few-shot examples teach the pattern concretely — split vs. not-split, mentioned vs. inferred, appropriate granularity.
- **B:** Hard count caps are arbitrary and can force the model to drop real skills or invent filler to hit the range.
- **C:** Post-processing can't fix inferred-but-not-mentioned skills or choose the right split for 'Python and SQL' — that decision has to be made at extraction time.
- **D:** Useful signal, but doesn't address the underlying inconsistency in how skills are parsed in the first place.

64 Your system has been operating with 100% human review for 3 months. Analysis shows that extractions with model confidence >90% have 97% accuracy overall. To reduce reviewer workload, you plan to automate high-confidence extractions. Before deploying, what validation step is most critical?

D4 · Prompt Engineering & Structured Output

- **A** Analyze accuracy by document type and field to verify high-confidence extractions perform consistently across all segments, not just in aggregate.
- **B** Compare accuracy at different confidence thresholds (85%, 90%, 95%) to find the optimal cutoff that maximizes automation while minimizing errors.
- **C** Run a two-week pilot routing 25% of high-confidence extractions directly to downstream systems and monitor error reports.
- **D** Verify that 97% accuracy meets requirements for all downstream systems that consume the extracted data.

Correct answer: A

EXPLANATION

Why it's correct: A. Correct. Aggregate accuracy hides segment failures — one document type could be 70% while others are 99%. Auto-routing by overall number alone risks systemic errors in the weak segments.

Option breakdown:

- **A:** Correct. Aggregate accuracy hides segment failures — one document type could be 70% while others are 99%. Auto-routing by overall number alone risks systemic errors in the weak segments.
- **B:** Threshold tuning is worth doing, but only after you know the accuracy holds uniformly across segments — otherwise you're optimizing a number that lies.
- **C:** A pilot is good practice, but it's downstream of the segment analysis — if a segment is systematically wrong, you'll learn it by harming users.
- **D:** An important product question, but it assumes the 97% holds everywhere — again, that's what segment analysis verifies first.

65 Your extraction pipeline processes contracts that frequently include amendments. When a contract contains both original terms and later amendments (e.g., original clause specifies "30-day payment terms" while Amendment 1 changes this to "45 days"), the model inconsistently extracts one value or the other with no indication of which applies. What's the most effective approach to improve extraction accuracy for documents with amendments?

D4 · Prompt Engineering & Structured Output

- **A** Redesign the schema so amended fields capture multiple values, each with source location and effective date.
- **B** Add prompt instructions to always extract the most recent amendment value and ignore superseded original terms.
- **C** Preprocess documents with a classifier that identifies and removes superseded sections before the main extraction step.
- **D** Implement post-extraction validation using pattern matching to detect amendments and flag those extractions for manual review.

Correct answer: A

EXPLANATION

Why it's correct: A. Correct. Amendments are structurally about versioned values. A schema with value + source location + effective date models the domain correctly and stops forcing the model to pick one.

Option breakdown:

- **A:** Correct. Amendments are structurally about versioned values. A schema with value + source location + effective date models the domain correctly and stops forcing the model to pick one.
- **B:** Downstream consumers sometimes need the original too (e.g., for dispute resolution). Hard-coding 'most recent wins' throws that away.
- **C:** Deleting sections with a classifier can accidentally strip content that's still legally effective — amendments often modify subsets of clauses.
- **D:** Routes every amended contract to a human — expensive, and still doesn't give downstream systems a structured answer.

66 Your extraction system implements automatic retries when validation fails. On each retry, the specific validation error is appended to the prompt. This retry-with-error-feedback approach resolves most failures within 2-3 attempts. For which failure pattern would additional retries be LEAST effective?

D4 · Prompt Engineering & Structured Output

A The model extracts keywords as a nested object organized by category when the schema requires a flat array of strings

B The model extracts citation counts as locale-formatted strings ("1,234") when the schema requires integers

C The model extracts dates as ISO 8601 datetime strings ("2023-03-15T00:00:00Z") when the schema requires only the date portion (YYYY-MM-DD)

D The model extracts "et al." for co-authors when the full list exists only in an external document not in the input

Correct answer: D

EXPLANATION

Why it's correct: D. Correct. No amount of retrying teaches the model information that isn't in the input. Retry-with-error-feedback only fixes mistakes the model could have gotten right from the source.

Option breakdown:

- **A:** A structural mismatch the model can definitely correct given feedback about the expected shape.
- **B:** The required number is present in the document — the model just needs to stop formatting it. Easy for feedback-retry to fix.
- **C:** Pure formatting fix — the underlying value is correct and the model can trim the time portion on retry.
- **D:** Correct. No amount of retrying teaches the model information that isn't in the input. Retry-with-error-feedback only fixes mistakes the model could have gotten right from the source.

67 Your extraction pipeline processes restaurant menus and must output structured JSON with fields for item names, descriptions, prices, and dietary tags. Some menus use inconsistent formatting—prices as "$12" vs "12.00", dietary info as icons vs text. What's the most reliable approach?

D4 · Prompt Engineering & Structured Output

A Use separate extraction calls for each field to ensure consistent handling of each type.

B Extract data as-is and normalize formats in post-processing code after Claude returns.

C Request multiple extraction attempts per document and select the most common format.

D Define a strict output schema and include format normalization rules in your prompt.

Correct answer: D

EXPLANATION

Why it's correct: D. Correct. Strict schema + explicit normalization rules ('prices as decimal with two places', 'dietary as enumerated tags') lets the model both extract and normalize in one pass.

Option breakdown:

- **A:** Multiple calls per menu multiplies cost and latency and loses cross-field context (e.g., which dietary icons sit next to which item).
- **B:** Post-processing a raw blob is brittle — deterministic code has to handle every format permutation. Better to let the model normalize at extraction time.
- **C:** Ensemble voting is expensive and can still pick a wrong-format majority.
- **D:** Correct. Strict schema + explicit normalization rules ('prices as decimal with two places', 'dietary as enumerated tags') lets the model both extract and normalize in one pass.

68 Your system extracts event metadata (date, location, organizer, attendee_count) from news articles using a JSON schema with all nullable fields. During evaluation, you observe the model frequently generates plausible but incorrect values for fields not mentioned in the article—for example, outputting "500" for attendee_count when the source contains no attendance information. What's the most effective way to reduce these false extractions?

D4 · Prompt Engineering & Structured Output

A. Add a post-processing step using a second LLM call to verify each extracted value exists in the source document.

B. Add prompt instructions to return null for any field where information is not directly stated in the source.

C. Make all schema fields required (non-nullable) with strict validation rules to ensure the model only outputs verifiable data.

D. Upgrade to a more capable model tier with improved instruction-following to reduce hallucination tendencies.

Correct answer: B

EXPLANATION

Why it's correct: B. Correct. The fields are already nullable; the model just needs an explicit instruction to prefer null over a plausible guess. This is the standard fix for schema-aware hallucination.

Option breakdown:

- **A:** Expensive second pass for a behavior you can correct at the source — tell the model to return null when the field isn't stated.
- **B:** Correct. The fields are already nullable; the model just needs an explicit instruction to prefer null over a plausible guess. This is the standard fix for schema-aware hallucination.
- **C:** Required fields force the model to invent values when information is missing — the opposite of what you want.
- **D:** Adds cost and still doesn't tell the model your policy — returning null instead of inventing.

69 After implementing tool use with strict schema definitions, JSON syntax errors are eliminated, but 5% of extractions still have valid JSON with empty arrays or null values for required fields like citations and methodology. Spot-checking reveals that source documents contain this information, but in varied formats—inline citations vs. bibliographies, methodology sections vs. details embedded in introductions. What's the most effective way to address these failures?

D4 · Prompt Engineering & Structured Output

A Implement retry logic that re-sends requests when validation detects empty required fields.

B Build a regex-based post-processing layer that scans source documents for citation patterns and methodology keywords, populating empty fields when the model fails to extract.

C Modify your schema to make citations and methodology optional, and flag incomplete records for manual review rather than failing validation.

D Add few-shot examples demonstrating extractions from documents with varied structures—showing how to identify citations in different formats and locate methodology details across section types.

Correct answer: D

EXPLANATION

Why it's correct: D. Correct. The failure mode is the model not recognizing varied formats. Concrete examples across the format distribution directly raise recall on the 5%.

Option breakdown:

- **A:** Retry on the same prompt won't suddenly teach the model to recognize methodology embedded in an intro. The extraction capability is the gap.
- **B:** Regex over prose is brittle, misses the nuanced cases (intro-embedded methodology), and ships silently wrong data.
- **C:** Papers over the symptom. The fields really are required downstream — routing them to humans trades one problem for reviewer workload.
- **D:** Correct. The failure mode is the model not recognizing varied formats. Concrete examples across the format distribution directly raise recall on the 5%.

70 Your extraction pipeline processes invoices and extracts line items, subtotals, tax amounts, and grand totals. During evaluation, you discover that in 18% of extractions, the sum of extracted line item amounts doesn't match the extracted grand total—sometimes due to OCR errors in the source document, sometimes due to extraction mistakes by the model. Downstream accounting systems reject records with mismatched totals. What's the most effective approach to improve extraction reliability?

D4 · Prompt Engineering & Structured Output

A Add a "calculated_total" field where the model sums extracted line items alongside a "stated_total" field. Flag records for human review when values differ.

B Extract line items and totals independently, then use a separate validation model to reconcile discrepancies by determining which extracted values are most likely correct.

C Add few-shot examples demonstrating invoices where extracted line items sum correctly to the stated total, encouraging the model to produce mathematically consistent extractions.

D Implement post-processing that automatically adjusts line item amounts proportionally when their sum doesn't match the stated total.

Correct answer: A

EXPLANATION

Why it's correct: A. Correct. Capturing both values makes the discrepancy a first-class signal — you catch OCR errors and extraction mistakes uniformly, and you can route only the mismatched 18% to humans.

Option breakdown:

- **A:** Correct. Capturing both values makes the discrepancy a first-class signal — you catch OCR errors and extraction mistakes uniformly, and you can route only the mismatched 18% to humans.
- **B:** A reconciliation model can mask OCR errors by silently picking one number over another — the accounting system still gets wrong data without a flag.
- **C:** Can nudge the model toward consistency, but when the source itself is internally inconsistent (OCR errors), the model has to choose one 'wrong'.
- **D:** Silent financial data rewrites are dangerous — you'd be fabricating line items for a downstream accounting system.

71 Your pipeline uses a tool called extract_metadata with a JSON schema for paper details. You've also defined lookup_citations and verify_doi tools for enrichment. During testing, you notice that when users include requests like "extract the metadata and tell me how cited it is," Claude sometimes calls lookup_citations first, which fails because it needs the DOI that extract_metadata would provide. What's the most effective way to ensure structured metadata extraction happens first?

D4 · Prompt Engineering & Structured Output

A Set tool_choice to "any" so Claude must use a tool, combined with system prompt instructions prioritizing extract_metadata.

B Set tool_choice to "auto" and reorder the tool definitions so extract_metadata appears first in the tools array, since Claude prioritizes earlier-listed tools.

C Set tool_choice to {"type": "tool", "name": "extract_metadata"} and process the enrichment requests in subsequent turns after receiving the extracted metadata.

D Set tool_choice to {"type": "tool", "name": "extract_metadata"} for every API call in the pipeline, ensuring Claude always extracts metadata before any enrichment can occur.

Correct answer: C

EXPLANATION

Why it's correct: C. Correct. tool_choice=specific-tool deterministically forces extract_metadata on the first turn. Then you hand control back to 'auto' to let the model use citations/DOI enrichment with the metadata in context.

Option breakdown:

- **A:** 'any' forces a tool call but doesn't force the right one. You'd still see lookup_citations picked first.
- **B:** There's no documented ordering preference to rely on; this assumes behavior that isn't contractual.
- **C:** Correct. tool_choice=specific-tool deterministically forces extract_metadata on the first turn. Then you hand control back to 'auto' to let the model use citations/DOI enrichment with the metadata in context.
- **D:** Pinning extract_metadata on every call prevents the enrichment tools from ever being used.

72 Your extraction uses tool use with a JSON schema where property_type is defined as an enum: ['house', 'apartment', 'condo', 'townhouse']. After deployment, 8% of extractions fail schema validation. Investigation reveals listings mention many uncommon property types—"studio", "loft", "duplex", "mobile home", "tiny house", "converted warehouse"—and new types continue appearing regularly. What's the most effective long-term solution?

D4 · Prompt Engineering & Structured Output

A Continuously expand the enum to include newly observed property types and add monitoring for additional edge cases.

B Add an "other" value to your enum with a separate property_type_detail string field for specifics when "other" is selected.

C Change property_type from an enum to a free-form string and implement a normalization step in post-processing.

D Add few-shot examples to your prompt demonstrating how to map unexpected property types to the closest existing enum value.

Correct answer: B

EXPLANATION

Why it's correct: B. Correct. Keeps the strong enum for the common cases (clean downstream joins) while giving a well-typed escape hatch that preserves detail. Stable long-term.

Option breakdown:

- **A:** Whack-a-mole enum maintenance — every new listing style risks another validation failure.
- **B:** Correct. Keeps the strong enum for the common cases (clean downstream joins) while giving a well-typed escape hatch that preserves detail. Stable long-term.
- **C:** Loses the validation guarantee entirely and pushes the normalization problem downstream with no canonical vocabulary.
- **D:** Forcing 'tiny house' into 'house' drops real distinctions downstream — you're silently lossy.

73 Your extraction system parses e-commerce product descriptions to extract specifications like dimensions, weight, and materials into JSON. Despite having a well-defined schema, the model inconsistently extracts the "materials" field—sometimes returning "cotton blend", other times "Cotton/Polyester mix", and occasionally omitting the field when material information is clearly present in the source. What's the most effective way to improve extraction consistency?

D4 · Prompt Engineering & Structured Output

A Make the "materials" field required instead of optional in the schema to force the model to always extract a value

B Switch to a more capable model tier since inconsistent extraction indicates insufficient model capability

C Set temperature to 0 to eliminate randomness and ensure deterministic outputs

D Add few-shot examples showing 2-3 complete input-output pairs with standardized material description formats

Correct answer: D

EXPLANATION

Why it's correct: D. Correct. Few-shot examples demonstrate the exact canonical format you want ('cotton/polyester' as a normalized list), and they also raise recall on material info that was being skipped.
Option breakdown:

- **A:** Required-ness doesn't solve format inconsistency, and it can force invented values when material info is genuinely missing.
- **B:** Costly, and doesn't address the actual cause — the model doesn't know the canonical output format you want.
- **C:** Determinism doesn't define the right format; it just makes the wrong-format answers consistent.
- **D:** Correct. Few-shot examples demonstrate the exact canonical format you want ('cotton/polyester' as a normalized list), and they also raise recall on material info that was being skipped.

74 After deployment, you find that 12% of extractions contain semantic errors that pass JSON schema validation (e.g., a duration like "30 minutes" incorrectly placed in an ingredient quantity field). Human reviewers have capacity to check only 20% of extractions. Which approach most effectively allocates reviewer attention?

D4 · Prompt Engineering & Structured Output

- **A** Have the model output field-level confidence scores, then calibrate review thresholds using a labeled validation set.
- **B** Randomly sample 20% of extractions for review, using corrections to track accuracy and identify error patterns.
- **C** Prioritize review of all extractions where required fields are empty or explicitly marked as not found.
- **D** Review all extractions from documents with formatting anomalies such as unusual layouts or mixed content types.

Correct answer: A

EXPLANATION

Why it's correct: A. Correct. Field-level confidence lets you route the low-confidence 20% — which is where the 12% semantic errors concentrate — to humans. Calibration makes the threshold choice data-driven.

Option breakdown:

- **A:** Correct. Field-level confidence lets you route the low-confidence 20% — which is where the 12% semantic errors concentrate — to humans. Calibration makes the threshold choice data-driven.
- **B:** Great for measurement, poor for coverage — random sampling only catches 20% of the 12% errors (~2.4% total). Confidence-based routing finds far more.
- **C:** Empty fields are a narrow error mode. The 12% problem is wrong values in populated fields, which this filter misses.
- **D:** A useful heuristic, but formatting anomaly doesn't reliably predict where fields got confused with each other.

05

DOMAIN 5

Context Management & Reliability

15% of scored content

3 questions

Context management, batch / async APIs, retries and reliability.

75 Your CI pipeline performs security-focused code reviews on approximately 50 PRs daily, currently costing $150/day using the synchronous API. Reviews are non-blocking—developers merge after tests pass and address findings in follow-up commits. You're evaluating the Message Batches API for its 50% cost reduction. What factor most determines whether batch processing is appropriate for this use case?

D5 · Context Management & Reliability

A. Whether you can structure each review as a single request without multi-turn refinement.

B. Whether reducing per-review latency from 30-60 seconds to near-instant matters for your workflow.

C. Whether review feedback arriving up to 24 hours after PR creation remains actionable.

D. Whether your result processing can handle reviews arriving in a different order than submitted.

Correct answer: C

EXPLANATION

The Message Batches API is asynchronous — most batches finish within 1 hour but can take up to 24 hours — so the decisive suitability factor is latency tolerance: whether review findings delivered up to 24 hours after PR creation are still actionable in a non-blocking, merge-then-fix workflow. Batching increases latency (B is backwards), results are correlated via custom_id so ordering is a non-issue (D), and single-request structuring (A) is secondary to the latency question.

76 Documents arrive continuously throughout business hours and need structured data extracted. To reduce costs, you want to use the Message Batches API (50% discount, up-to-24-hour processing window). Your SLA specifies that extraction results must be available within 30 hours of document arrival with 99.9% reliability. Which batching strategy is most appropriate?

D5 · Context Management & Reliability

A Submit batches every 6 hours containing documents from that window

B Submit a single batch at end of day containing all documents from that day

C Submit batches every 4 hours containing documents from that window

D Use the real-time API for all documents instead of batch processing

Correct answer: C

EXPLANATION

Why it's correct: C. Correct. Max 4-hour wait + up to 24-hour batch SLO = 28-hour worst case, leaving a 2-hour cushion under the 30-hour SLA to absorb batch variance and hit 99.9%.

Option breakdown:

- **A:** Worst-case a document waits 6 hours to be batched and then up to 24 hours to process = 30 hours exactly. No safety margin for a 99.9% reliability target.
- **B:** A doc arriving just after morning startup waits all day to be batched plus up to 24 hours to process — well over 30 hours.
- **C:** Correct. Max 4-hour wait + up to 24-hour batch SLO = 28-hour worst case, leaving a 2-hour cushion under the 30-hour SLA to absorb batch variance and hit 99.9%.
- **D:** Meets SLA trivially but throws away the 50% cost saving the question asks you to capture.

77 After your daily batch of 10,000 documents completes, 300 documents (3%) failed with "context_length_exceeded" errors. The results file identifies each failure by custom_id. What's the most cost-effective approach to process these failures?

D5 · Context Management & Reliability

A Reprocess the entire batch with prompt caching enabled to reduce the cost of retrying requests with identical system prompts

B Resubmit only the 300 failed documents after chunking them into smaller pieces, then combine the partial extractions

C Resubmit the entire 10,000 document batch using a model tier with a larger context window

D Increase the max_tokens parameter for the 300 failed documents and resubmit them in a new batch

Correct answer: B

EXPLANATION

Why it's correct: B. Correct. Targeted, and it addresses the actual cause (input too long). Chunk the oversized docs, extract per chunk, then merge — minimum tokens, fixes the specific failure mode.

Option breakdown:

- **A:** Re-running 10,000 successful documents to solve 300 failures is wasteful regardless of caching savings. And caching doesn't fix the length-exceeded error.
- **B:** Correct. Targeted, and it addresses the actual cause (input too long). Chunk the oversized docs, extract per chunk, then merge — minimum tokens, fixes the specific failure mode.
- **C:** Expensive (reprocessing 9,700 successes) and may still fail on truly outsized documents.
- **D:** max_tokens controls output length, not input length. It doesn't solve context_length_exceeded, which is about the combined input-plus-output budget.

Quick Answer Key

77 questions

1 **A**	2 **D**	3 **A**	4 **C**	5 **D**	6 **C**
7 **D**	8 **A**	9 **C**	10 **D**	11 **C**	12 **B**
13 **D**	14 **A**	15 **A**	16 **D**	17 **C**	18 **D**
19 **B**	20 **C**	21 **A**	22 **C**	23 **B**	24 **C**
25 **C**	26 **B**	27 **A**	28 **B**	29 **A**	30 **B**
31 **C**	32 **C**	33 **A**	34 **C**	35 **A**	36 **C**
37 **A**	38 **B**	39 **C**	40 **B**	41 **A**	42 **A**
43 **A**	44 **C**	45 **B**	46 **D**	47 **C**	48 **A**
49 **C**	50 **D**	51 **B**	52 **C**	53 **A**	54 **D**
55 **B**	56 **B**	57 **D**	58 **D**	59 **C**	60 **C**
61 **D**	62 **D**	63 **A**	64 **A**	65 **A**	66 **D**
67 **D**	68 **B**	69 **D**	70 **A**	71 **C**	72 **B**
73 **D**	74 **A**	75 **C**	76 **C**	77 **B**	