60 Claude Prompts — the full pack

10 categories, 6 prompts each, copy-paste ready. You commented "60" — here's every single one.

Before you paste anything

Three rules, then you're set:

- **Fill the [PLACEHOLDERS].** Every bracket like [TOPIC] or [DATASET] is where your real thing goes. The more specific you make it, the better the answer.
- **One prompt, one job.** Don't stack three of these into one message. Run one, get the result, then run the next.
- **Start with today's task.** Don't read this top to bottom. Jump to the category that matches what's on your plate right now and run one prompt from it.

That's it. The pack is a toolbox, not a course.

1/10 — Learning Anything Fast

Make Claude teach you in layers, quizzes, and drills instead of dumping a wall of text.

Prompt 1

```
Explain [TOPIC] to me in four layers, each building on the last: (1) to a 10-year-old using one everyday analogy, (2) to a smart beginner, covering how it actually works, (3) to an expert, with terminology, edge cases, and failure modes, (4) a one-sentence core. After layer 4, list the 3 ideas most people get wrong.
```

Prompt 2

```
Be my Socratic tutor on [TOPIC]. Ask me one question at a time, starting easy and getting harder, and wait for my answer before the next. If I'm wrong or vague, give a hint, not the answer, and let me retry. After 8 questions, score my grasp 1–10 and name the exact gaps to review next.
```

Prompt 3

```
I want to learn [TOPIC]. Build me a spaced-repetition plan over 14 days. For each session give the date, what to re-explain from memory, and a different audience to teach it to that day (a child, a skeptic, a peer). End with 5 recall questions I should be able to answer cold by day 14.
```

Prompt 4

```
I want to get better at [SKILL] fast. Break it into 5-7 sub-skills, then tell me the single one with the highest leverage for a beginner and why. Design a 15-minute daily deliberate-practice drill for it: the exact reps, one measurable target, and the mistake to watch for. Nothing generic.
```

Prompt 5

```
Build me a skill tree to go from zero to competent at [SKILL]. Give 4 stages (beginner to advanced), and for each: 3 concepts to master, one project that proves it, and a checkpoint test to pass before leveling up. Flag any prerequisite I must not skip, and estimate hours per stage.
```

Prompt 6

```
I'll explain [CONCEPT] to you in my own words as if you're new to it. Read my explanation, then act as a tough examiner: point out every place I was vague, wrong, or hand-waved, rank the gaps by how badly they'd trip me up, and give me one sharper analogy for the weakest part. Here's my explanation: [YOUR EXPLANATION]
```

2/10 — Research & Analysis

Turn piles of sources and long documents into decisions, not summaries.

Prompt 1

```
Synthesize these sources on [TOPIC], don't just summarize each. Format as: (1) points they agree on, (2) where they directly conflict and why, (3) 3-4 themes cutting across them, (4) the biggest gap none of them address. Cite which source backs each claim. Sources: [PASTE SOURCES]
```

Prompt 2

```
Summarize [DOCUMENT] for a busy decision-maker. Format as: a one-sentence bottom line, 3 key findings (one line each), the strongest supporting evidence, the biggest caveat or unknown, and one action it implies. Skip background and filler. Flag anything the document asserts without evidence.
```

Prompt 3

```
Extract the signal from this [TRANSCRIPT/THREAD/REPORT]. List the 5 most important claims ranked by how much they'd change my decision, each in one line. Then separately list what's just filler, repetition, or opinion stated as fact. End with the one question I still need answered. Source: [PASTE]
```

Prompt 4

```
Stress-test my position: [YOUR CLAIM]. Give me three voices,
labeled clearly: an Advocate making the strongest case for it, a
Skeptic building the best steelman against it with real
evidence, and a Neutral judge who weighs both and states which
side is stronger and what would change the verdict.
```

Prompt 5

```
Fact-check each claim in this text. For every claim, give a
verdict (Supported / Disputed / Unverifiable), a confidence of
High, Medium, or Low, the reasoning, and what evidence would
change your mind. Do not invent sources; if you can't verify,
say so plainly. Format as a table. Text: [PASTE]
```

Prompt 6

```
Compare these options on [DECISION]: [OPTION A], [OPTION B],
[OPTION C]. Build a decision matrix scoring each across the 5
criteria that actually matter for my goal of [GOAL], with a one-
line reason per cell. Below the table, name the best pick, the
runner-up, and the one condition that would flip your answer.
```

3/10 — AI Agents & MCP

For anyone building agents: the design decisions, tools, evals, and error recovery — before you write code.

Prompt 1

```
Decide agent vs workflow for [TASK]. Score it on step predictability, need for dynamic tool selection, and cost of a wrong action. Return: recommended pattern (single prompt / prompt-chain / router / orchestrator-workers / autonomous agent), one-line reason, the simplest version that ships, and the failure mode that would force the next tier up.
```

Prompt 2

```
Design the tool set for an agent doing [WORKFLOW]. List 3-6 high-impact tools only (not one per API endpoint). For each: name, one-sentence description, JSON Schema inputs, and what the response returns. Make responses token-efficient -- return IDs and summaries, not full dumps -- and flag any tool whose output could blow the context window.
```

Prompt 3

```
Write the MCP server surface for [DOMAIN]. Split capabilities into tools (actions the model calls), resources (read-only context the client loads), and prompts (reusable templates). Give each tool a JSON Schema, a clear description, and a structured error shape. State which are read vs write, and which need user confirmation before executing.
```

Prompt 4

```
Design an orchestrator-subagent system for [GOAL]. Define the lead agent's job (plan, delegate, synthesize), 2-4 specialized subagents with a one-line scope each, what context each receives, and how results merge. Note which subagents run in parallel vs sequentially, and the token/latency budget per subagent.
```

Prompt 5

```
Build an eval harness for my agent on [TASK]. Give me 8-10 test cases spanning happy path, ambiguous input, and tool failure. For each: input, expected tool-call sequence, and a pass rule. Track total tool calls, tokens, runtime, and tool-error rate. Output a table plus the single metric that best signals regressions.
```

Prompt 6

```
Add error recovery and memory to an agent doing [WORKFLOW]. Specify: which tool errors are retryable (with backoff) vs fatal, how the agent reports being stuck instead of looping, what state persists across steps vs gets summarized, and a hard stop after N failed actions. Return the recovery policy as a short decision table.
```

4/10 — Prompt Engineering

Prompts that fix your other prompts: clearer, tested, harder to break.

Prompt 1

```
Rewrite [PROMPT] to be clear and direct, as if briefing a new hire with no context. Spell out the goal, audience, and exact steps in order, and state what to do with edge cases. Return the rewritten prompt only, then a short bullet list of every ambiguity you removed so I can confirm the intent is still mine.
```

Prompt 2

```
Improve [TASK] using multishot prompting. Generate 3 diverse, high-quality examples wrapped in XML: put each inside <example> with <input> and <ideal_output> tags, covering one normal case, one edge case, and one tricky case. Keep examples consistent in format so Claude pattern-matches the structure, not just the content.
```

Prompt 3

```
Add chain-of-thought to [PROMPT] for a task needing real
reasoning. Instruct the model to work step by step inside
<thinking> tags before answering, then give the final answer
inside <answer> tags. Do not let it skip the thinking. Return
the upgraded prompt and one line on why separating reasoning
from output improves accuracy here.
```

Prompt 4

```
Turn [TASK] into a prompt-chaining pipeline instead of one mega-
prompt. Break it into 3-4 focused subtasks, each its own prompt,
where every step passes its result to the next inside XML tags.
For each link give: input, single responsibility, output tag.
Explain where a self-check or verification step should sit in
the chain.
```

Prompt 5

```
Design an eval for prompt [PROMPT]. Define success criteria that
are measurable, then produce 8-10 test cases as a table: input |
ideal_output | what_it_probes (edge case, format, refusal,
hallucination). Recommend the grading method (exact match, code
assertion, or LLM-as-judge) per case and how I would score a
pass rate.
```

Prompt 6

```
Harden [PROMPT] against hallucination using Anthropic's methods.
Explicitly permit 'I don't know' when evidence is missing,
require every factual claim to cite a supporting quote from
[SOURCE], and for long documents instruct the model to extract
relevant word-for-word quotes FIRST, then answer only from
those. Return the revised prompt.
```

5/10 — Career & Interviews

Resume, interview practice, salary negotiation, LinkedIn — the whole job hunt in six prompts.

Prompt 1

```
You are an ATS resume expert. Compare my resume against this job
post: [JOB DESCRIPTION]. [PASTE RESUME]. List the top 15 exact-
phrase keywords from the post I am missing, mark which are hard
requirements, then rewrite 4 bullet points to embed them
naturally with metrics. End with an ATS match score out of 100
and 3 fixes to raise it.
```

Prompt 2

```
Act as an interviewer for a [ROLE] role. Ask me one behavioral question at a time (leadership, conflict, failure, initiative). After each answer, score it 1-10 on the STAR method (Situation, Task, Action, Result), flag any missing part, and give one sharper rewrite. Keep Action at 60% of the answer. Do 5 questions total.
```

Prompt 3

```
You are a compensation negotiator. The offer is [OFFER] for [ROLE] in [CITY]; my target is [TARGET]. Using anchoring and a range-based counter, write a 4-line email that opens with enthusiasm, cites market data, states a range 10-20% above the offer, and asks. Then give 3 replies to 'that is our max' that keep it warm.
```

Prompt 4

```
Write a 3-paragraph cover letter for this role: [JOB DESCRIPTION], using my background: [BACKGROUND]. Open with a quantified achievement that mirrors their top requirement (no restating my resume). Middle: 2 proof points tied to their needs. Close: company-fit line plus a clear call to action. Under 300 words, confident, no cliches.
```

Prompt 5

```
Optimize my LinkedIn for recruiter search as a [ROLE]. Given
[CURRENT HEADLINE] and [ABOUT], write: a 150-character headline
with role, specialty, and value; a first 3 lines of About that
hook before 'see more'; and the 5 exact-phrase skills to weave
in for semantic search. Explain each choice in one line.
```

Prompt 6

```
Build a 30-60-90 day plan for my new [ROLE] job; context:
[TEAM/GOALS]. Return a table with 3 phases: Days 1-30 Learn, 31-
60 Contribute, 61-90 Own. Each phase: 3 SMART goals, one
measurable early win, and who to build a relationship with. Add
5 smart questions to ask my manager in week one.
```

6/10 — Web & SEO

Get found on Google without hiring an agency: keywords, audits, copy, speed, local.

Prompt 1

```
I run [BUSINESS] and want to get found on Google. Suggest 20 keywords real customers would search for [TOPIC], each with the likely search intent (informational, commercial, or transactional) and a content idea to target it. Return as a table sorted from easiest to rank to hardest, and flag the 5 best to start with.
```

Prompt 2

```
Act as an SEO auditor. Here is the page: [PASTE URL OR TEXT]. Check it against on-page basics: title tag, meta description, one clear H1, logical heading order, keyword and intent match for [KEYWORD], readability, and internal links. Return a table of issue, why it matters, and the exact fix, ranked by impact.
```

Prompt 3

```
Write 5 title tag and meta description options for my page about [PAGE TOPIC] targeting [KEYWORD]. Keep each title under 60 characters with the keyword near the front, and each description 150–160 characters that earns the click without keyword stuffing. Return as a numbered table with character counts shown.
```

Prompt 4

```
Rewrite my homepage copy for [BUSINESS], which helps [AUDIENCE] achieve [OUTCOME]. Lead with a headline that answers 'what's in it for me' in 5 seconds, sell benefits over features, and end with one action-verb CTA. Give me a headline, subheadline, 3 benefit bullets, and a button label, plus 2 alternate headlines.
```

Prompt 5

```
My site [URL] feels slow and fails Core Web Vitals. Explain LCP, INP, and CLS in plain English with their good thresholds (LCP under 2.5s, INP under 200ms, CLS under 0.1), then give me a prioritized checklist of fixes for a non-developer, like compressing images to WebP, deferring scripts, and setting image dimensions.
```

Prompt 6

```
I own a local [BUSINESS TYPE] in [CITY] and want to rank in Google Maps. Audit my Google Business Profile priorities: best primary and secondary categories, every field to fill, a photo plan, and a simple system to earn and reply to reviews. Return a checklist grouped by 'do today', 'this week', and 'ongoing'.
```

7/10 — Automation & Workflows

For n8n / Make / Zapier builds that survive real traffic, not just the demo run.

Prompt 1

```
Design an n8n/Make/Zapier workflow triggered by [TRIGGER] that performs [GOAL]. Output: (1) trigger + auth, (2) numbered node sequence with the app each step calls, (3) exact field mapping between steps as source -> destination, (4) filter/branch conditions, (5) final action. Flag any step likely to hit rate limits.
```

Prompt 2

```
Add production-grade error handling to my workflow that calls [API]. Give me: a retry policy with exponential backoff (1s, 2s, 4s + jitter, cap 3-5 attempts), which HTTP codes to retry vs fail fast, a dead-letter path for permanent failures, and a dedicated error workflow that alerts [CHANNEL] with the failed payload.
```

Prompt 3

```
Build an idempotency gate for my webhook [TRIGGER] so provider retries never double-process. Output: how to derive the idempotency key (header or body hash), the guard step that checks a seen-store (Redis/Postgres, 24-48h TTL), respond-200-fast then process async, and HMAC signature verification to reject spoofed calls.
```

Prompt 4

```
Add a human-in-the-loop approval step before my AI agent runs [SENSITIVE_ACTION]. Output: which tools to gate vs leave open, the Slack/email approval message showing tool name + AI-chosen params + reasoning, an editable field for the reviewer, and a timeout fallback (auto-approve low-risk / auto-reject high-risk / escalate).
```

Prompt 5

```
Chain these API calls to go from [INPUT] to [OUTPUT]: list the call order, what field from each response feeds the next request, how to loop cursor/offset pagination until done, where to normalize mismatched fields between [TOOL_A] and [TOOL_B], and where to cache to stay under rate limits. Return as a step table.
```

Prompt 6

```
Spec an AI-agent-in-a-workflow for [TASK]. Output: the trigger, the system prompt defining the agent's job and boundaries, the exact tools it can call with input schemas, guardrails (max steps, banned actions, cost cap), the approval gate for irreversible steps, and how the final result is written back to [DESTINATION].
```

8/10 — API Design

Design APIs other developers don't curse at: errors, idempotency, pagination, auth, webhooks.

Prompt 1

```
Review [ENDPOINT] against REST conventions. Check: noun-based resource paths, correct verb + status code mapping, consistent snake_case or camelCase, pagination on list responses, and a versioning strategy (URI /v1 vs header). Output a table of issue, why it matters, and the corrected signature. Flag anything that breaks backward compatibility.
```

Prompt 2

```
Design a machine-readable error taxonomy for [API] using RFC 9457 problem+json. Give 8-10 error types with a stable type URI, title, HTTP status, and detail. Mark each retryable or not (4xx except 408/429 = no, 429 and 5xx = yes). Include an example body and a rule for when to expose internal detail vs redact it.
```

Prompt 3

```
Make [POST_ENDPOINT] idempotent. Specify the Idempotency-Key header contract: client generates a UUID, server caches the response keyed on it with a 24h TTL, and duplicate keys return the stored result without re-running side effects. Define behavior on key reuse with a different body (409) and on in-flight retries. Return the request/response flow.
```

Prompt 4

```
Convert [LIST_ENDPOINT] from offset to cursor pagination. Use an opaque cursor tied to the last item so concurrent inserts don't skip or duplicate rows. Return the response shape (data, next_cursor, has_more), the query change, page-size limits with a default, and how the client loops until has_more is false. Note the tradeoff vs random page jumps.
```

Prompt 5

```
Design the auth and rate-limit layer for [API]. Pick an OAuth2
flow (client_credentials vs authorization_code + PKCE) with
reasoning, JWT claims and expiry, and scope granularity. Add
rate limits returning 429 with Retry-After plus X-RateLimit
headers, and client-side exponential backoff with jitter. Output
the headers and a limit table by tier.
```

Prompt 6

```
Design webhooks for [EVENT_TYPE]. Cover: event payload schema
with a stable id and type, HMAC-SHA256 signature header for
verification, a timestamp to reject replays, a retry schedule
with backoff, and consumer idempotency on the event id. Specify
what a 2xx vs non-2xx from the consumer means, and how to expose
a redelivery endpoint.
```

9/10 — DevOps & Deploy

Paste your Dockerfile, manifests, pipeline, or Terraform and get a real review back.

Prompt 1

```
Optimize this Dockerfile for size, build speed, and security: [DOCKERFILE]. Convert to a multi-stage build (build stage + slim runtime), order layers so dependency installs cache before source copy, pin the base image by digest, run as a non-root USER, and add a .dockerignore. Return the rewritten Dockerfile with inline comments and a before/after estimated image size.
```

Prompt 2

```
Review these Kubernetes manifests: [MANIFESTS]. Check for missing resource requests/limits, absent liveness/readiness/startup probes (liveness restarts, readiness gates traffic, startup covers slow boot), no securityContext (runAsNonRoot, readOnlyRootFilesystem), and missing PodDisruptionBudget. Output: Issue | Severity | Fixed YAML snippet.
```

Prompt 3

```
Audit this CI/CD pipeline config: [PIPELINE]. Flag slow or missing stages and reorder as lint -> unit -> build -> integration -> deploy with fail-fast and caching. Verify no plaintext secrets, per-environment scoping, least-privilege deploy identity, and required-status gates on main. Output: Stage | Problem | Fix | Est. time saved. Give the corrected pipeline file.
```

Prompt 4

```
Review this Terraform for correctness, drift, and security:
[TERRAFORM]. Flag hardcoded secrets, unpinned providers/modules,
missing remote state + locking, public exposure (open SGs,
public buckets), and no encryption. Map findings to what TFLint,
Checkov/Trivy, or an OPA policy would catch. Output: Resource |
Risk | Severity | Fix (HCL). End with the plan you'd expect.
```

Prompt 5

```
Define SLOs and burn-rate alerts for service [SERVICE] with
these SLIs: [SLIS]. For each (availability, latency p99, error
rate) set a target, a 28-day error budget, and multi-window
burn-rate alerts (fast 1h + slow 6h) so paging fires only on
real budget burn. Output: SLI | SLO target | Alert threshold |
PromQL expression. Note what should page vs ticket.
```

Prompt 6

```
Write an incident runbook for failure mode [FAILURE] on service
[SERVICE]. Structure it: Symptoms & alerts | Impact/blast radius
| Diagnosis (exact commands + dashboards) | Mitigation (rollback
or feature-flag kill) | Escalation & on-call | Verification. Add
a zero-downtime rollback (blue-green switch or canary halt).
Keep every step copy-pasteable.
```

10/10 — Data & Analytics

From messy dataset to a defensible decision: EDA, cleaning, SQL, sanity checks, dashboards.

Prompt 1

```
Run first-pass EDA on [DATASET]. Give me pandas for: shape + df.info() dtypes, df.describe() and value_counts() on key columns, missing-value counts per column, duplicate rows, and 3 obvious anomalies or outliers worth investigating. End with 5 questions this data can answer and any column I should not trust yet.
```

Prompt 2

```
Write a pandas cleaning pipeline for messy [DATASET]. Handle in order: coerce dtypes (pd.to_datetime, numeric), strip/lowercase text, standardize categories, decide drop vs fillna per column with a stated reason, remove exact + near-duplicates, and rename columns to snake_case. Output runnable code plus a before/after row-count log.
```

Prompt 3

```
Write SQL for a cohort retention table on [TABLE]: group users by their [SIGNUP_PERIOD] cohort, then show the % still active in each following period. State the cohort definition, the return-event that counts as retained, and how you handle partial/incomplete cohorts. Return the query plus how to read the triangle.
```

Prompt 4

```
Analyze funnel dropoff for [FUNNEL_STEPS] using [TABLE]. Give me SQL counting distinct users at each step, step-to-step conversion %, the biggest dropoff, and a segment breakdown by [DIMENSION]. Then list 3 hypotheses for the worst leak and the one metric to watch after a fix. Flag any step ordering assumptions.
```

Prompt 5

```
I see [RESULT] in [METRIC]. Before I act, run sanity checks: is the sample large enough, is the difference statistically significant (state test + p-value), could a confounder like [VARIABLE] or seasonality explain it, is it survivor/selection bias, and is the metric definition stable. Output: verdict (real / noise / unclear) + what to check next.
```

Prompt 6

```
Design a decision dashboard for [GOAL] / [TEAM]. Pick one North Star metric, then 4–6 supporting metrics pairing each leading indicator next to the lagging one it drives, plus 1–2 guardrail metrics. For each: definition, why it's actionable not vanity, and the threshold that should trigger a decision. Return as a table.
```

Where this fits

Prompts like these are the first level: you ask better, you get better answers. The level above that is when the prompts stop living in a chat window and start living inside a system that's wired into your real files and tools and runs on its own. That's the work I do all day at Axionox, and every system I build started as a prompt like the ones in this pack.

Your next step

Pick the one category that matches what you're working on today. Run one prompt from it, filled in properly, right now — it takes two minutes and you'll feel the difference on the first answer.

Prompt pack curated from @leadgenman's collection.

Built by Axionox. I'm Saad — I build real AI systems for a living. If you want to see what these prompts look like grown up into a working system, reply to the DM and I'll show you one. Follow for more like this.