

Visible Reasoning Is Not a Universal Optimizer: Persona- and Thinking-Dependent Effects in Analytics Code Generation

Bhawani Shankar Leelar
Optum AI

bhawani.leelar@optum.com

Pawan Chorasiya
Optum AI

pawan.chorasiya@optum.com

Davin Hill
Optum AI

davin.hill@optum.com

Robert E. Tillman
Optum AI

rob.tillman@optum.com

Tamer Soliman
Optum AI

tamer.soliman@optum.com

Abstract

Visible Chain-of-Thought (CoT) is often treated as a broadly useful reasoning instruction, yet analytics code generation combines natural-language ambiguity, schema grounding, target-language constraints, and model-specific inference behavior. Because the same analytics request can be expressed in two distinct target languages-SQL and Python (pandas)-this setting provides a natural test of a common but under-examined assumption: that visible reasoning is more effective when its representation matches the requested target, as in “think in SQL” or “think in Python.” Together with generic instructions such as “think step-by-step,” such recommendations remain insufficiently evaluated under controlled, execution-based comparisons. We study a query-matched SQL-pandas benchmark that crosses persona phrasing, target language, visible-CoT format, control prefixes, direct generation, and internal-reasoning configurations. The results do not support either a universal accuracy advantage from visible CoT or a consistent benefit from matching the reasoning representation to the target language. Instead, the effects depend on the persona, target, model configuration, and internal-reasoning setting. The control ablations further distinguish effects of reasoning content from those of prompt format. These findings indicate that reasoning strategies should be selected jointly for the model, persona, target, and internal-reasoning configuration rather than adopted as universal defaults. More broadly, the study provides a controlled framework for identifying when visible reasoning improves executable generation, when it primarily perturbs model behavior, and when the internal-reasoning configuration is the more consequential factor.

1 Introduction

Modern language models increasingly translate natural-language data requests into executable analytics code. The same request can be realized in two different

paradigms: Structured Query Language (SQL), which specifies relational operations declaratively, and Python (pandas), which expresses the result through a sequence of DataFrame transformations. Visible CoT is now a common prompting device for both, whether as a general instruction to reason step-by-step (Wei et al., 2022) or as a target-matched instruction such as *think in SQL* or *think in Python*. If the most useful visible-CoT representation is the one that matches the requested code, it should preferentially benefit procedural pandas generation. A query-matched SQL-pandas benchmark makes this proposition testable: the database task, schema, question intent, and executable reference result can be fixed while the requested target and visible-CoT representation vary.

Several alternative mechanisms complicate this target-matching hypothesis. A visible CoT may alter generated code without supplying a better task solution: code-like notation can prime target vocabulary, a fixed-step output contract can alter decoding behavior, and any additional prefix changes the context presented to the model. A fluent rationale is not necessarily a faithful account of the computation leading to the final answer (Turpin et al., 2023; Lanham et al., 2023); additional inference-time computation is also not uniformly useful across tasks (Snell et al., 2025). We therefore distinguish two channels. In this paper, *visible CoT* (or *visible reasoning*) denotes text in the externally observable prompt-response exchange: a condition-specific CoT instruction or prefix and the decoded **thinking** list returned with the final code. The explicit-CoT conditions request eight labeled entries; direct generation and the controls have different contracts described in Section 3. This visible text is not taken to reveal latent computation. *Internal reasoning* denotes unobserved model- or service-level inference-time computation. The provider-exposed switches studied here are therefore termed *internal-reasoning configurations*; we neither observe nor interpret provider-side internal traces (Zelikman et al., 2024; Hao et al., 2025). Thus, toggling an internal-reasoning configuration changes an

inference setting while we continue to measure only visible output traces and executable code.

We examine these channels on a query-matched SQL-pandas (Li et al., 2023; Hill et al., 2026) panel that crosses persona phrasing, target language, visible CoT format, direct generation, three control prefixes, and internal-reasoning configurations. The controls are ablations of coherent NL CoT, not candidate prompting methods: length filler removes task-relevant semantics while retaining approximate prefix volume; scrambling disrupts word order while retaining related lexical material; and wrong content supplies a reasoning-like prefix from a different task. Each is compared with both direct generation and coherent NL CoT. This dual comparison separates a change caused by loss of task-aligned reasoning from a change caused by adding a long, structured context.

The central result is not a universal visible-CoT or target-matching advantage. Instead, outcomes vary by persona, target, model configuration, and internal-reasoning setting. In particular, the ablations show that reasoning content and prompt format need not have the same behavioral role: a scrambled prefix can match or exceed coherent NL CoT in localized cells even though it does not preserve a chronological solution plan. Visible CoT also adds token and latency overhead whose return is configuration-specific. The relevant unit of comparison is consequently a model-persona-target configuration, not a global prompting default. In summary, this work makes four contributions:

- We present a query-matched evaluation that holds the database task and executable reference result fixed while varying SQL or pandas targets, persona phrasing, direct generation, visible-CoT format, and model configuration.
- We test visible NL-CoT with three complementary controls: length filler, scrambled tokens, and wrong content. Each control is compared with both coherent NL-CoT and direct generation to separate task alignment from structured-prefix effects.
- We find that neither visible CoT nor target-language matching provides a universal accuracy advantage. The observed changes instead depend on the persona, target, and model configuration, with a localized scrambled-prefix result motivating a prompt-format explanation rather than a new default.
- We examine visible CoT and internal-reasoning configurations alongside difficulty, token use, and latency. These analyses show why direct generation remains an essential reference when selecting a reasoning configuration.

The next Section situates this design within prior work on visible CoT, code-generation intermediates, and inference-time computation before specifying the matched experiment.

2 Related Work

Visible CoT and causal interpretation. CoT prompting established that decoded intermediate text can improve performance on multi-step tasks (Wei et al., 2022). In prior work, such text may be supplied in demonstrations, elicited as an output, or used alongside unreported model-side computation; it should not automatically be read as an internal reasoning trace. What an externally visible rationale contributes remains less settled. Ablations of demonstrations and noisy rationales indicate that relevance, ordering, and task fit can matter separately from local plausibility (Wang et al., 2023; Zhou et al., 2024). Other analyses document settings in which an incomplete or misleading trace harms performance (Zheng et al., 2025; Su et al., 2026). Faithfulness work further cautions that a fluent explanation may not be the causal computation responsible for the final answer (Turpin et al., 2023; Lanham et al., 2023). Our length-filler, scrambled, and wrong-content conditions instantiate these concerns as dual-baseline ablations of a visible NL-CoT prefix and output contract for executable analytics generation.

Reasoning representations for code. Several code-generation approaches use plans, programs, or structured intermediate representations to decompose a task. Program-aided language models delegate suitable sub-computations to an executable program (Gao et al., 2023), while work on intermediate languages and code prompting compares natural-language, pseudocode, and code-like representations (Deng et al., 2024; Sun et al., 2024; Puerto et al., 2024). Recent studies extend this line to multilingual structured traces (Jin et al., 2025b) and to broad comparisons of CoT strategies in programming benchmarks (Jin et al., 2025a). These contributions seek more effective generation procedures and may use reasoning representations differently from our prompt-response trace. In contrast, our objective is diagnostic: the task, reasoning-slot structure, and evaluator are held fixed while the externally visible representation, target language, and user phrasing are crossed.

Text-to-SQL planning and analytics generation. Reasoning is also central to text-to-SQL systems, commonly through schema linking, query planning, verification, or repair. SQL-of-Thought, for example, combines staged planning and correction in a multi-agent procedure (Chaturvedi et al., 2025); STaR-SQL uses reasoning-oriented learning and verification for text-to-SQL (He et al., 2025). Such systems change several ingredients jointly, including the number of calls, feedback, and selection policy. Our factorial design does not estimate the value of full agentic systems with extended planning, tool use, or open-ended multi-turn revision. Instead, it isolates whether the surface form of an otherwise matched visible prefix changes SQL or pandas execution accuracy under

a bounded recovery policy.

Internal computation and adaptive inference.

Quiet and latent reasoning approaches distinguish model-side computation from the public token stream (Zelikman et al., 2024; Hao et al., 2025), and test-time-compute work shows that additional inference effort should be considered jointly with task difficulty and resource cost (Snell et al., 2025). In code generation, uncertainty-guided CoT similarly allocates extended reasoning selectively rather than unconditionally (Zhu et al., 2025). These approaches motivate our separate treatment of visible CoT and internal-reasoning configurations, as well as the difficulty-conditioned token and latency analyses. They do not license an equivalence between a longer visible trace and provider-side internal reasoning; our configuration comparisons observe behavioral consequences, not the content of internal computation.

Benchmarks, schema grounding, and persona.

Text-to-SQL benchmarks such as Spider and BIRD foreground cross-domain schema grounding and execution correctness (Yu et al., 2018; Li et al., 2023); data-science benchmarks demonstrate the additional challenges of realistic Python-library code (Lai et al., 2023; Ouyang et al., 2025). The matched setting used here combines database-grounded requests and executable SQL and pandas targets derived from BIRD (Li et al., 2023) with Original, Novice, and Expert persona phrasings supplied by BIRD-Persona (Hill et al., 2026). The paired text-to-Python/text-to-SQL setting makes persona a particularly important moderator, because the same underlying intent can leave different amounts of procedural and schema detail implicit (Hu et al., 2026; Ding et al., 2026). We therefore analyze wording as part of the experimental design rather than averaging it away.

Taken together, this literature motivates a matched evaluation that separates reasoning representation, task phrasing, and inference configuration. The next Section defines that design and the execution-based estimands used to evaluate it.

3 Experimental Design

This Section defines the matched factors, common output contract, and execution-based estimands used throughout the paper. It first specifies what is held constant across conditions, then describes the model configurations and the paired statistical comparisons that support the Results in Section 4.

3.1 Matched Panel and Conditions

From the 1,534-task BIRD-Persona benchmark (Hill et al., 2026), seeded random sampling retained at most 10 questions from each database-difficulty stratum. After the

pre-inference exclusions documented in Appendix A.3, 304 database-grounded questions form the common panel for all six completed model configurations. For every question, database schema and executable gold answer are fixed while phrasing (Original, Novice, Expert), target (SQL, pandas), and reasoning condition vary. The four explicit CoT conditions share the same eight semantic slots. The representation contract, not the decomposition, is manipulated within those four conditions.

NL. Complete each slot with concise English prose and avoid formal code bodies.

SQL. Use clause, projection, source, join, predicate, aggregate, subquery, alias, and validation fragments.

Python. Use literals, assignments, selectors, masks, pandas transformations, and assertions.

Pseudocode. Use an uppercase structured vocabulary including OUTPUT, MAP, TABLES, COLUMNS, COMPUTE, FILTER, BUILD, CONVERT, and VALIDATE.

Direct. Receive no visible-CoT definition and return an empty visible thinking list.

Three ablations of coherent NL-CoT alter properties of its visible prefix:

Length filler. Ablate task-relevant NL CoT sentences while retaining approximately matched textual volume in an eight-label unrelated prefix.

Scrambled. Ablate chronological order by permuting the words of a reasoning-like eight-step template while retaining its lexical material.

Wrong content. Ablate task relevance by supplying a reasoning trace associated with a different query matched by database/difficulty where possible.

Thus each model configuration contains $304 \times 3 \times 2 \times 8 = 14,592$ matched trials. Appendix A.2 specifies the prompt slots, controls, query exclusions, evaluator, and full condition matrix.

Every trial uses a shared schema-grounded JSON response contract containing a visible `thinking` list, target-language code, and an explicit code-type field. The four explicit-CoT prompts require exactly eight labeled steps with the same semantic roles and differ only in the required representation; direct generation requires an empty visible thinking list. The controls retain the shared code and schema guardrails but replace the coherent NL-CoT prefix with the relevant ablation. Thus, the exact-eight-step statistic reported in the Results is an observed response-compliance measure, not a substitute for execution accuracy or a measurement of latent reasoning. Internal-reasoning configurations are toggled separately; they change the inference setting, while the study continues to measure only visible output traces and executable

Model	Full evaluated model and configuration
GLM4	GLM-4-32B-0414; non-reasoning
Qwen3	Qwen3-30B-A3B-Instruct-2507; non-reasoning
Gemma4-off	Gemma 4 26B-A4B-IT; internal reasoning disabled
Gemma4-on	Gemma 4 26B-A4B-IT; internal reasoning enabled
Gemini Flash-0	Gemini 2.5 Flash; internal-reasoning budget 0
Gemini Flash-24k	Gemini 2.5 Flash; internal-reasoning budget 24,576

Table 1: Six completed configurations used in the experiments.

code. Full prompt and control construction details are given in Appendix A.2.

The task panel supplies a database-grounded request, a schema context, executable SQL and pandas reference programs, and three semantically matched persona phrasings for the same analytic intent (Hill et al., 2026). Original is the benchmark wording from (Li et al., 2023); Novice uses a less technical user-facing formulation; Expert makes analytic details more explicit. We retain the benchmark’s paired formulation rather than treating these as different questions. Consequently, a within-query comparison isolates a change in prompt wording or reasoning condition while holding the database, reference output, and target language fixed. This is especially important for SQL-pandas comparison: the same analytic result can be declarative in SQL yet require an explicit sequence of DataFrame operations in pandas.

Appendix A.1 defines the benchmark-derived annotations used in the adjusted analyses: BIRD difficulty, an AmbiSQL-derived balanced ambiguity score (Ding et al., 2026), a SQL-grounded phrasing-specificity score, and Structural Complexity Score (SCS), an equal-weight SQL-structure count.

The 304 questions are present for every completed model configuration and every factorial cell. The common frame prevents a model-specific missing row from changing the query distribution used in a paired contrast; Appendix A.3 reports the pre-comparison audit that produced this frame. Table 1 gives the full names and establishes the compact reference names used in the remainder of the paper.

The open-weight configurations were served with vLLM on an AWS g6.12xlarge instance using BF16 weights and a 32,768-token context limit. Gemini Flash was queried through the provider API from an Apple M4 Max client with 64 GB unified memory; provider-side inference hardware was not observable. Full infrastructure details, including completion limits, retries, timeouts, soft-

ware dependencies, and the execution environment, are reported in Appendix A.4. These serving paths motivate within-configuration, rather than cross-provider, resource comparisons.

3.2 Evaluation and Inference

Generated code is executed in an isolated worker and matched to a freshly executed reference result after deterministic normalization. Parse failures, runtime failures, and missing outputs are incorrect. The primary estimand is the paired within-query accuracy change relative to direct generation. Exact two-sided McNemar tests, 4,000 query-cluster bootstrap intervals, and Benjamini-Hochberg (BH) correction evaluate 252 primary model-persona-target-condition contrasts. Declared secondary families cover internal-reasoning configuration contrasts, target matching, persona interactions, and the control mechanism tests. A complementary query-clustered logistic Generalized Estimating Equations (GEE) adjusts for target, BIRD difficulty, database, AmbiSQL-derived ambiguity (Ding et al., 2026), SQL-grounded phrasing specificity, and SCS (Hu et al., 2026); exact paired contrasts remain primary. Complete procedures and all correction families are reported in Appendix C. Each completed cell uses one zero-temperature generation under the fixed experiment seed; the analysis therefore estimates conditional behavior of these six serving configurations, not sampling variability over repeated stochastic decodes.

For a reasoning condition m and direct generation 0, the paired effect is the query-average difference

$$\hat{\Delta}_m = \frac{1}{n} \sum_{q=1}^n (y_{qm} - y_{q0}),$$

reported in percentage points. Here $n = 304$, q indexes the retained query, m is the reasoning condition being compared with direct generation, 0 denotes direct generation, and $y_{qm} \in \{0, 1\}$ is the exact-execution correctness indicator for query q under condition m . Exact McNemar tests use only discordant query pairs and are appropriate because both conditions are evaluated on the same underlying task. The 95% intervals resample complete query clusters rather than individual rows, preserving the dependence induced by the two targets, three phrasings, and eight conditions attached to each query. BH correction controls the declared family of 252 model-persona-target-condition tests. The same-family internal-reasoning, target-language matching, persona interaction, and control-mechanism questions have separate, explicitly reported correction families because they answer different estimands.

The complementary GEE is deliberately not a replacement for paired inference. It asks whether a condition association persists after including model configuration, target, persona, difficulty, database, ambiguity, phrasing specificity, and SCS, with query-clustered robust covariance. This model supplies a check on confounding

Inference family	Tests	BH discoveries
Mode-direct	252	0
Target matching	36	0
Internal-reasoning toggle	96	16
Persona interactions	756	1

Table 2: Multiplicity-controlled inference families. Primary condition tests are the first row; other rows address distinct predeclared questions.

by annotated query properties; it does not turn correlated benchmark features into independently randomized causes. Token and latency summaries are similarly descriptive. They retain individual CoT formats, condition on difficulty, and compare values within a serving configuration rather than asserting that latency is directly comparable across infrastructure.

This design preserves query-level pairing while making the scope of each comparison explicit. The next Section reports the execution accuracy, ablation, internal-reasoning, and resource findings.

4 Results

This Section first establishes direct generation as the execution reference, then evaluates visible-CoT formats, ablations, internal-reasoning configurations, and resource distributions against that reference. The paired-inference design from Section 3 is retained in every comparison.

4.1 Direct Generation as the Reference

Table 3 gives the corresponding descriptive results for Original phrasing. Direct generation is an essential part of this comparison rather than a residual baseline: its accuracy differs substantially by target and model configuration. Across the six configurations, direct SQL accuracy exceeds direct Python accuracy by 8.22-16.45 percentage points, and direct generation ties the strongest direct-or-explicit-CoT value for Gemma4-off on Python. The four explicit formats do not produce a common ranking across SQL and Python. Together, these cells establish a target- and configuration-dependent reference against which the paired tests below should be read.

4.2 No Universal Visible-CoT Advantage

No condition survives the 252-test primary family, and no target-matched format has a BH-corrected advantage (Table 2). The direct-reference matrix above is therefore descriptive evidence of heterogeneity, not a replacement for the corrected paired tests.

Persona nevertheless matters. The paired persona-interaction family in Table 2 and the all-persona GEE

both retain mode-by-persona evidence after BH correction, indicating that phrasing is not a uniform shift in task difficulty. SCS does not retain a distinct adjusted association in that model, whereas ambiguity and phrasing specificity do; these covariates describe the benchmark rather than causal persona mechanisms. The appendix supplies the complete persona-by-difficulty matrices, which show why the Original-focused main table must not be generalized to every phrasing regime. It also shows that a control can be the highest cell in a row, particularly under Novice wording. This pattern motivates the dual-baseline ablation analysis below instead of a post-hoc selection of the highest-scoring prefix.

The paired accuracy comparisons must be read with their resource distribution in view. Figures 2, 3, and the appendix latency analysis retain target, difficulty, and individual CoT formats: the internal-reasoning view uses signed enabled-minus-disabled resource buckets, whereas the token view locates direct and visible-CoT accuracy in model-adaptive bins. These views show that visible CoT usually occupies larger token and latency budgets, but the associated accuracy return is small and configuration-specific rather than systematic. Harder queries shift both direct and prompted generations toward larger budgets, and enabling internal reasoning changes the resource range occupied by paired evaluations. Thus, the results do not support visible CoT as a generally cost-effective default; they support configuration-specific selection against a direct-generation reference.

4.3 Controls as NL-CoT Ablations

The controls test semantic relevance, order, and task alignment in a visible prefix; they are diagnostic ablations, not optimized prompts. Table 4 compares each control with both coherent Natural Language (NL) CoT and direct generation. Length filler and wrong content show no corrected difference from either comparator. Scrambling shows no corrected difference from coherent NL CoT, but is modestly higher than direct generation. This does not imply that scrambled reasoning is useful. Rather, it suggests that a related but non-chronological prefix can sometimes alter the model’s generation regime without preserving a valid stepwise plan.

The full matrix shows that this effect is persona- and difficulty-dependent, with the strongest localized result under Novice-SQL. Figure 1 and the appendix compliance tests show why the effect should not be attributed to recovery of the eight-step contract: scrambled outputs have low exact-step adherence, and compliance is not associated with correctness within the scrambled control. The supported interpretation is therefore a configuration-specific sensitivity to perturbed task-related context, not a general benefit of incoherent reasoning.

Model	Target	Direct	CoT					Control		
			NL	SQL	Python	Pseudocode	mean	Filler	Scrambled	Wrong
GLM4	SQL	23.68	28.29	28.29	25.99	29.28	27.96	24.01	25.66	27.30
GLM4	Python	15.13	19.74	18.42	17.11	17.76	18.26	16.45	15.13	15.79
Qwen3	SQL	29.93	33.55	30.26	33.55	30.59	31.99	31.58	34.54	30.26
Qwen3	Python	21.71	22.04	21.38	20.39	20.72	21.13	23.03	25.33	24.01
Gemma4-off	SQL	36.18	37.17	37.83	37.50	36.51	37.25	37.17	36.84	36.51
Gemma4-off	Python	19.74	17.76	19.41	19.74	18.09	18.75	18.42	18.42	18.75
Gemma4-on	SQL	37.83	37.83	38.16	38.49	39.14	38.40	39.14	39.47	41.12
Gemma4-on	Python	26.32	27.30	27.63	26.32	25.66	26.73	24.34	25.99	25.99
Gemini Flash-0	SQL	35.86	36.51	37.17	35.53	39.80	37.25	39.47	39.14	41.12
Gemini Flash-0	Python	23.03	25.00	25.99	22.04	26.32	24.84	26.97	25.00	24.34
Gemini Flash-24k	SQL	36.84	37.50	35.86	37.50	36.51	36.84	36.18	36.51	33.22
Gemini Flash-24k	Python	26.32	25.00	27.63	25.99	26.32	26.23	26.64	27.63	23.36

Table 3: Original-phrasing exact execution accuracy (%) along with the mean of the four explicit-CoT formats. In each row, bold marks the best direct-or-explicit-CoT value and the best control value.

Ablation	Δ NL	Δ direct	BH outcome (NL; direct)
Length filler	-0.36	0.90	not detected; not detected
Scrambled	0.16	1.43	not detected; higher
Wrong content	-0.49	0.77	not detected; not detected

Table 4: Original-phrasing control contrasts, pooled over six configurations, targets, and matched queries. Δ NL is control minus NL-CoT accuracy; Δ direct is control minus direct-generation accuracy. Changes are percentage points. “Not detected” denotes failure to reject after BH correction; “higher” is an observed corrected contrast.

4.4 Internal-Reasoning Configuration and Resource-Bounded Accuracy

The signed resource plots in Figure 2 provide the model-family detail behind the preceding overview. The paired points occupy different resource ranges for Gemma4 and Gemini Flash, and their accuracy changes differ across targets and difficulty levels. Thus, enabling internal reasoning redistributes evaluations across resource budgets as well as changing accuracy; it cannot be interpreted as a common treatment across model families. The complete three-persona display is provided in the appendix.

Visible CoT must also be evaluated against its resource cost. Figure 3 keeps the four formats separate and conditions on target and difficulty. Direct and explicit-CoT points are distributed across model-adaptive token bins rather than along a shared accuracy curve: harder queries shift both target languages toward larger budgets, while visible CoT generally extends that spread. The relative positions of direct and CoT points differ by model and target, and latency view (in the appendix) should be compared only within a serving configuration. These budget-stratified summaries are descriptive rather than causal dose-response plots or cross-provider latency com-

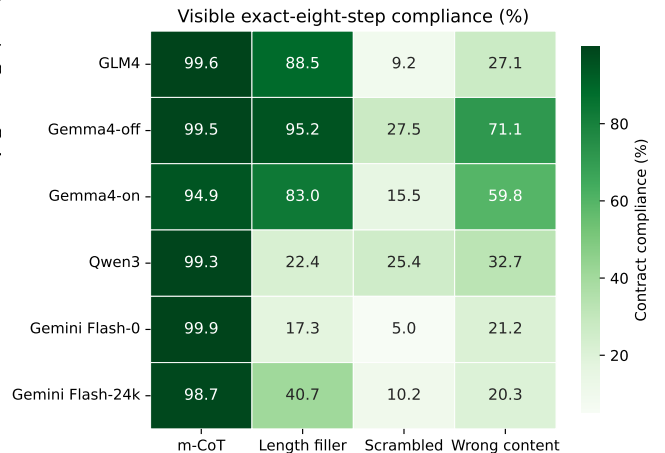


Figure 1: Exact eight-step compliance by model. *m*-CoT averages the four explicit-CoT formats.

parisons.

These results distinguish the observed patterns from their causal interpretation. The next Section considers the mechanism hypotheses and the resulting evaluation guidance.

5 Discussion

The results favor a conditional account of visible CoT. Direct generation is not uniformly dominated: Table 3 shows that it can match the strongest direct-or-explicit-CoT option, while other cells favor different explicit formats. The primary analysis therefore does not support either a universal accuracy benefit or a target-language-matched advantage for the tested panel. Persona is not an ornamental factor: the adjusted interaction and the complete appendix matrices show that phrasing changes

Thinking-toggle accuracy changes by signed resource bucket: Original phrasing

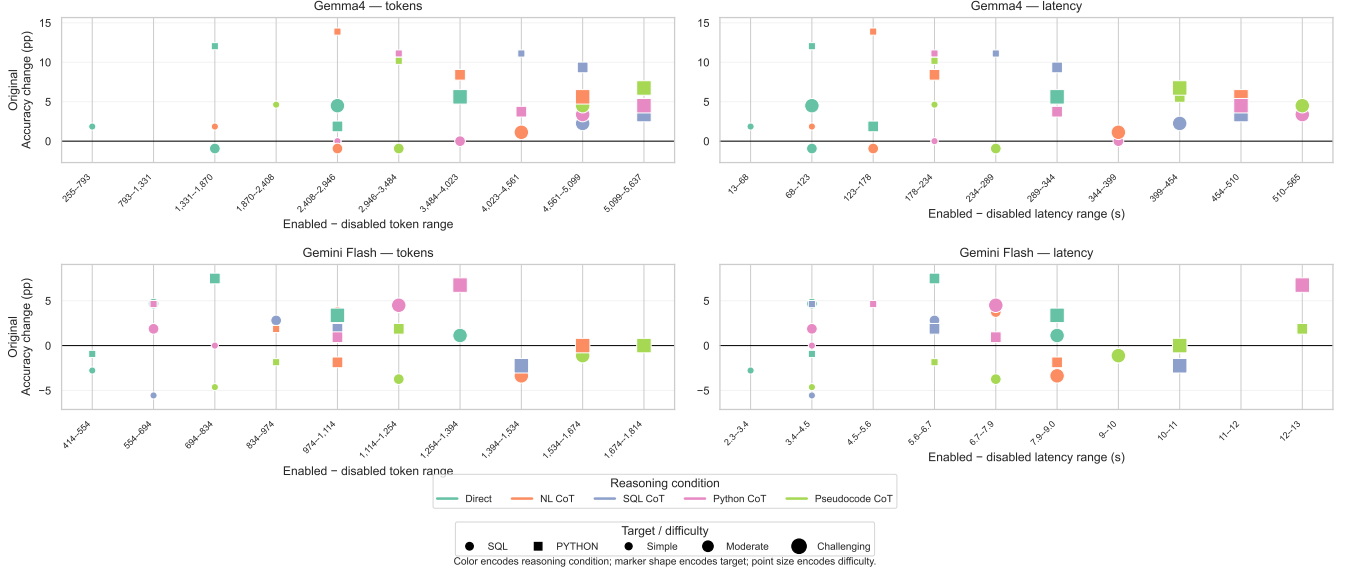


Figure 2: Original-phrasing internal-reasoning configuration effects within family-adaptive signed token and latency buckets. Each point is the mean paired enabled-minus-disabled accuracy change in its resource bucket.

Accuracy within model-adaptive tokens bins: Original phrasing

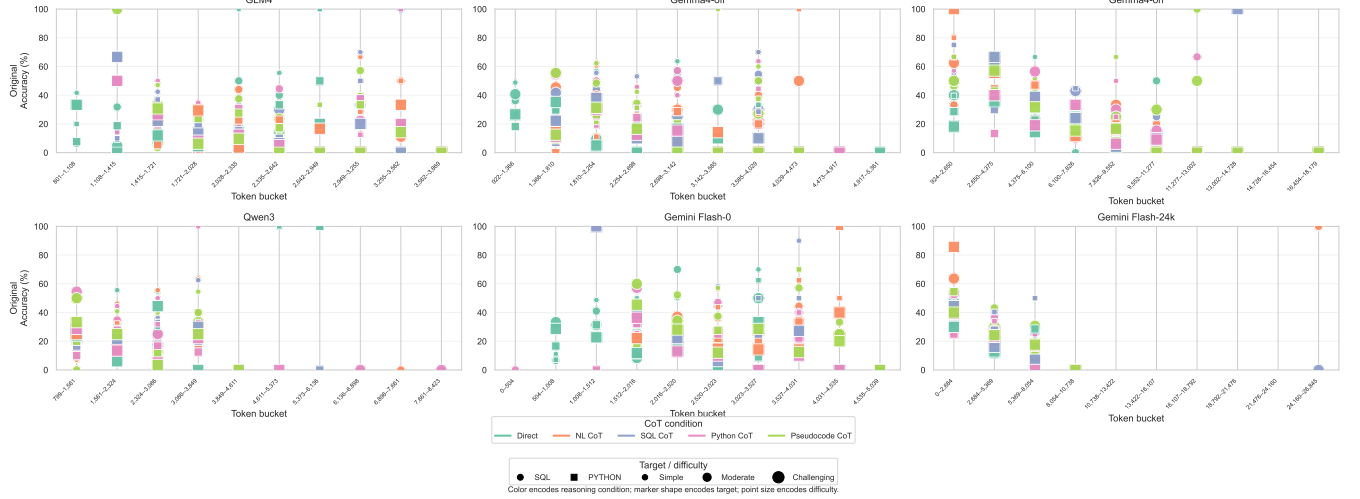


Figure 3: Original-phrasing accuracy within model-adaptive total-token bins. Each point is the mean accuracy in a bin.

the operating regime in which a prompt is evaluated.

The ablations sharpen the interpretation. Their results do not support the claim that a longer prefix alone explains accuracy movements, nor do they show that models reliably follow an injected wrong plan. Instead, the surprising scrambled outcome is consistent with an interaction between task-related tokens and a visible output structure. This is a falsifiable mechanism hypothesis: a frozen-trace experiment can independently manipulate token content, sequence, and step format to determine which component carries the effect.

For deployment, the relevant comparison is an accuracy-resource frontier, not a single accuracy-per-token ratio. Figures 2, 3, and the appendix latency analysis show that visible CoT usually occupies larger token and latency budgets, while its accuracy gain is small and configuration-specific. Difficulty also shifts both direct and prompted generations toward larger budgets, and internal-reasoning settings change the resource range of paired evaluations. These bucket views support configuration-specific selection against direct generation, not a global recommendation to add visible CoT or enable

internal reasoning.

The next Section consolidates these conditional findings and their implications for evaluating reasoning prompts in analytics-code generation.

6 Conclusion

In matched analytics-code generation trials, direct generation remains a competitive, configuration-dependent reference rather than a uniformly weaker alternative to visible CoT. Visible CoT is neither a universal optimizer nor a reliably target-language-matched one; its effects vary with persona, target, model configuration, and internal-reasoning configuration. The dual-baseline ablations indicate that prompt-format and lexical effects can perturb generation independently of coherent visible CoT, while the difficulty-conditioned resource analyses show why configuration-specific choices are necessary. Together, these findings provide a controlled basis for future work on when visible CoT helps code generation, when it changes behavior without improving it, and when an internal-reasoning configuration is the more consequential lever.

7 Limitations

The controls are diagnostic ablations of coherent NL-CoT, not counterfactual edits of one frozen rationale. They preserve different properties of an injected prefix and therefore cannot separately identify the causal roles of semantic content, sequence, length, and output structure. The wrong-content and scrambled prefixes provide reasoning-like contexts, not interventions on latent computation; the scrambled-control findings should be read as mechanism hypotheses about sensitivity to perturbed task-related context, not as evidence that incoherent reasoning is generally beneficial.

Each matched cell contributes one temperature-zero evaluation under a fixed bounded-recovery policy. Because the runner permits limited recovery after transient call failures, JSON-parse failures, and execution failures, the outcome reflects a controlled single-task evaluation loop rather than either a pure first-attempt sample or an unrestricted agentic system. Repeated decoding seeds, service replicas, and independently sampled wrong-content traces would be needed to quantify stochastic and operational variance. Persona phrasing, ambiguity, phrasing specificity, difficulty, and SCS are correlated benchmark descriptors, so the adjusted GEE associations should not be read as independent causal effects.

The six configurations provide structured coverage, not a randomized model sample. Same-family internal-reasoning contrasts are informative, but architecture, training data, weights, access path, serving stack, and provider implementation remain confounded across families. Token fields may be provider-reported or locally es-

timated, and latency is client-observed wall-clock time rather than a hardware-normalized service-time metric; resource analyses therefore support within-configuration comparisons against direct generation, not cross-provider cost benchmarks.

Finally, the task scope is executable SQL and pandas analytics generation under a shared JSON response contract. The conclusions may not transfer to open-ended software engineering, interactive tool-using agents, alternative execution engines, multi-turn repair systems, or substantially different reasoning interfaces.

A Factorial and Artifact Audit

The common analysis frame contains 87,552 unique factorial cells: six model configurations, 304 query IDs, three personas, two targets, and eight reasoning conditions. Every configuration contributes 14,592 rows. There are no duplicated setting-query-persona-target-condition keys, no missing final-correctness labels, and no missing SCS values in the evaluated frame. The internal-reasoning-enabled configurations use the same cached native gold executions and common question set as their paired disabled or budget-zero configurations. Table 6 summarizes the completed common-frame inventory.

This appendix records the construction details needed to audit the annotations, exclusions, controls, serving settings, and analyses.

A.1 Annotation score definitions

The adjusted analyses use three benchmark-derived annotations: phrasing specificity, ambiguity, and SQL structural complexity. They are not additional experimental treatments. They are fixed query-level or phrasing-level descriptors used to audit whether a reasoning-condition comparison is confounded with properties inherited from the BIRD-Persona benchmark panel.

Phrasing specificity. Phrasing specificity is the benchmark-provided wording annotation used in the adjusted analyses to reflect how explicitly a question wording states schema-grounded details relative to its paired SQL task (Hill et al., 2026). We treat it only as a fixed inherited benchmark descriptor of the displayed wording, not as an experimental treatment or a stand-alone measure of semantic equivalence.

Ambiguity. Ambiguity is derived from an AmbiSQL-style ambiguity detector and taxonomy (Ding et al., 2026). Let L_x be the multiset of level-2 ambiguity labels detected for wording x , and let K be the number of level-2 taxonomy labels ($K = 7$ in the detector used here). The coverage and volume components are

$$A_{\text{cov}}(x) = \frac{|\text{unique}(L_x)|}{K}, \quad A_{\text{vol}}(x) = \frac{|L_x|}{|L_x| + K}.$$

The balanced ambiguity score used as the GEE covariate is

$$A_{\text{bal}}(x) = \lambda A_{\text{cov}}(x) + (1 - \lambda) A_{\text{vol}}(x),$$

with $\lambda = 0.5$. Thus the score lies in $[0, 1]$, increases when more ambiguity categories are detected, and also gives diminishing credit for multiple detected ambiguity items. It is a study-specific metric derived from AmbiSQL outputs, not a claim about AmbiSQL’s original evaluation protocol.

Structural Complexity Score. SCS is an equal-weight count of SQL structural features computed from the executable reference SQL after parsing it in the SQLite dialect. Following the matched text-to-Python/text-to-SQL benchmark’s use of SCS (Hu et al., 2026) and related text-to-SQL workload profiling practice (Ma et al., 2025), let c_q^{col} , c_q^{tab} , c_q^{where} , c_q^{join} , c_q^{agg} , c_q^{scalar} , c_q^{cte} , and c_q^{subq} denote the counts of distinct referenced columns, physical tables, atomic **WHERE** predicates, join pairs, aggregate calls, nonaggregate scalar-function calls, common-table-expression definitions, and nested subqueries for query q . Then

$$\text{SCS}_q = c_q^{\text{col}} + c_q^{\text{tab}} + c_q^{\text{where}} + c_q^{\text{join}} + c_q^{\text{agg}} + c_q^{\text{scalar}} + c_q^{\text{cte}} + c_q^{\text{subq}}.$$

The score characterizes the reference SQL structure. A generated SQL or pandas program can be correct without reproducing the same structure.

The following subsections distinguish the instructions imposed on a model from response properties measured after generation.

A.2 Prompt construction and compliance protocol

The main paper specifies the shared three-field JSON response and eight semantic slots. At implementation level, SQL must be one self-contained SQLite query. Pandas code executes in a supplied DataFrame environment, cannot import modules or access files, networks, or processes, and must assign the answer to **result**. Schema identifiers must come from the displayed context. These target-specific guardrails are identical across reasoning conditions.

The three controls replace the coherent NL-CoT prefix while retaining the same schema, target-code, and JSON contracts. Length filler is a seeded, unrelated eight-step trace assembled from an everyday-task phrase bank. Its text is extended until its count under the experiment’s shared token counter approximately matches a deterministic eight-step reasoning-like template for the current query and target. Scrambling applies a seeded permutation to the whitespace-delimited tokens of that template, preserving its token multiset but removing sequence and its visible step labels. Wrong content selects a different

query with the same database identifier and difficulty when available, using a seed determined by the query identifier. It injects that source query’s pre-generated NL-CoT trace from a fixed bank; if no bank entry is available, it uses a deterministic non-gold eight-step trace derived only from the source question, database identifier, and target. Neither path exposes reference SQL, reference pandas code, or execution evidence to the target trial. For the retained study bank, source-query NL-CoT traces were generated offline with **gpt-5-mini** (bank prompt version v4) and stored only as thinking-step text plus source-query metadata; evidence fields, gold SQL, and gold pandas code were excluded from the bank entries.

We distinguish instruction compliance from correctness. For explicit CoT, the eight-string requirement is part of the prompt. For controls, the injected prefix preserves, destroys, or replaces visible step labels but does not add the explicit CoT conditions’ exact-eight-output instruction. Exact eight-step output is therefore an observed behavioral measure rather than a uniform control requirement. We parse each response against the shared JSON contract and record the number of generated visible thinking steps. Exact step compliance is the indicator that the parsed thinking list contains eight strings. The main-paper heatmap aggregates this indicator by model and reports explicit-CoT as its mean over the four formats; the full format-level matrix is reported in Figure 8. Execution correctness remains the primary outcome throughout.

A.3 Reproducibility protocol

The seeded database-difficulty sampling described in the main paper produced 310 candidates. Before inference, five were excluded because native SQL gold answers were not executable and one because its required function was unsupported by the SQLite evaluator. Exclusions were fixed before condition comparisons and did not depend on model output. Intersecting completed settings then yielded the audited 304-query common frame in Table 6.

For each query, the database identifier, difficulty label, schema context, question phrasing, target language, reasoning prefix, generated answer, execution status, normalized output, correctness label, token counts, and latency were retained. The gold answer was executed once per database and target under the same deterministic evaluator used for model outputs. A model output was marked correct only when its executed result matched the normalized gold result; parse failures, runtime failures, and missing outputs were marked incorrect rather than silently discarded. No semantic judge was used for the primary label.

Query order was fixed by the common manifest, and all summaries followed key and missingness audits. Reported or locally estimated token fields were retained: provider/serving-stack usage metadata was preferred when present, with a common local counter used as

fallback. Latency was measured locally with a monotonic, high-resolution clock around the client request/response path and accumulated with the timed trial phases; it is not supplied by a provider. These measurements support within-configuration runtime descriptions rather than a hardware-normalized comparison across serving paths.

Generation used a fixed experiment seed (20260630) and temperature zero for the completed runs. Each matched cell contributed one recorded generation; the analysis does not average repeated decoding seeds. Internal-reasoning configurations are identified in Table 6 and the main model table. They are request or service configurations that may alter model-side inference, but no provider-side thought tokens or internal-reasoning traces were requested, retained, or analyzed; every configuration is evaluated through the same visible response contract and executable code. Model-specific token budgets and provider-side internal-reasoning controls are therefore treated as serving configurations rather than as architecture isolations. Authentication, transport retries, and provider response metadata were handled before the final row-level audit; no failed request was silently replaced by a second sampled question.

No reasoning format, correction family, or resource-bin width was selected by maximizing accuracy. The model panel is descriptive coverage of completed configurations, not a hyperparameter search.

A.4 Serving and runtime configuration

Table 5 records the execution environment for the six completed configurations. The open-weight configurations were served with vLLM on an AWS `g6.12xlarge` instance provisioned with 48 vCPUs, 192 GiB memory, and four NVIDIA L4 GPUs (96 GB aggregate GPU memory). Inference used BF16 weights and a 32,768-token maximum context length. The internal-reasoning switch for Gemma4 was passed at request time, so Gemma4-off and Gemma4-on use the same checkpoint while differing in the enabled inference channel. The Gemini Flash configurations were queried through the provider API from a Mac with an M4 Max processor and 64 GB unified memory. This computer was an API client and experiment orchestrator; it was not the provider’s inference host. Accordingly, resource totals compare conditions within a serving configuration and should not be read as a hardware-normalized benchmark between the two serving paths.

The experiment configuration fixes temperature at zero, the experiment seed at 20260630, a request timeout of 600 seconds, at most two runner-level recovery attempts after the initial generation (shared across transient-call, JSON-parse, and execution repair), and a 30-second execution timeout. Non-thinking vLLM generations use a 4,096-token completion limit. Gemma4-on starts at 8,192 tokens and, only when the completion reaches that limit, retries the same prompt at 12,288 and then 16,384 tokens subject to the 32,768-token con-

Evaluation path	Runtime configuration
GLM4; Qwen3	vLLM on AWS <code>g6.12xlarge</code> (4 L4 GPUs, 48 vCPUs, 192 GiB); BF16; 32,768-token context; no internal-reasoning setting.
Gemma4-off/on	Same AWS/vLLM host and checkpoint; BF16; 32,768-token context; request-level internal reasoning disabled/enabled.
Gemini Flash-0/24k	Provider API, issued from Mac M4 Max with 64 GB unified memory; provider-side serving hardware not observed.
All configurations	Temperature 0; fixed experiment seed; bounded repair and execution; deterministic reference normalization.

Table 5: Serving and execution configuration. Full model identifiers and the compact reference names used elsewhere are listed in the main model table.

Model	Rows	Queries	Correct (%)
GLM4	14,592	304	37.38
Qwen3	14,592	304	41.18
Gemma4-off	14,592	304	43.54
Gemma4-on	14,592	304	46.57
Gemini Flash-0	14,592	304	46.15
Gemini Flash-24k	14,592	304	45.96

Table 6: Completed common-frame inventory.

text bound. Gemini Flash uses provider-side internal-reasoning budgets of 0 and 24,576; no thought summaries are requested, and the provider’s final-output limit is not overridden. The analysis environment used Python 3.12 with pandas, SQLGlott, matplotlib, seaborn, statsmodels, and vLLM 0.25.1. The original row-level records do not preserve a complete operating-system image, CUDA/driver image, or provider-side hardware manifest. Each latency is client-observed wall-clock time: it includes the instrumented request/response call and other timed trial phases, rather than an internal provider service-time field. It is therefore a within-configuration outcome.

Open-model servers used a bounded context window to retain key-value-cache headroom during batched serving. Generated programs were evaluated in an isolated worker with deterministic normalization before comparison against a fresh native reference execution. Per-request timeout, transport retry, parsing, execution status, token fields, and locally measured end-to-end latency were retained in the row-level artifact. A failed generation contributes an incorrect outcome rather than causing a query to be resampled.

With the shared panel and runtime inventory fixed, the next section expands the main paper’s Original-only display to every persona and target. As throughout the paper, failure to reject a paired difference is not evidence of equivalence because no equivalence margin was predeclared.

B Detailed Persona-Wise Accuracy

Table 7 complements the Original-phrasing matrix in the main paper with every persona, target, and visible-CoT condition. It preserves direct generation as an explicit comparator and shows the phrasing-dependent cells that are necessarily compressed in the main narrative.

The matrix supplies the descriptive cells for the paired contrasts below; it does not itself establish a condition effect. In particular, direct generation remains competitive in multiple rows, so a highlighted control or CoT cell is not evidence of a universal prompting advantage. The next section specifies the tests, confidence intervals, multiplicity families, and adjusted covariates.

C Paired Statistical Procedures

C.1 Notation and analysis unit

Let $q \in \{1, \dots, Q\}$ index retained queries, with $Q = 304$ in the common frame. Let s index the six model configurations, p the persona phrasing, t the target language, and r the reasoning condition. The binary outcome is

$$y_{qspr} = \begin{cases} 1, & \text{correct,} \\ 0, & \text{otherwise.} \end{cases}$$

Here correct means that the generated program executes and matches the gold result after deterministic normalization. All paired contrasts hold q , s , p , and t fixed and vary only the reasoning condition. For the primary family, the comparator a is direct generation. For the control-mechanism family, a can be direct generation, NL CoT, another control, or the explicit-CoT mean, depending on the stated hypothesis.

C.2 McNemar Test

For a reasoning condition r and comparator a , let

$$b = \sum_q I(y_{qspr} = 1, y_{qspta} = 0),$$

$$c = \sum_q I(y_{qspr} = 0, y_{qspta} = 1),$$

where $I(\cdot)$ is the indicator function. Thus b counts matched queries on which condition r is correct and comparator a is wrong, while c counts the opposite discordance.

Under the paired null, the $b + c$ discordant outcomes are exchangeable with probability one half (McNemar, 1947). Let $B \sim \text{Binomial}(b + c, 0.5)$, $P_- = \Pr[B \leq \min(b, c)]$, and $P_+ = \Pr[B \geq \max(b, c)]$. The exact two-sided McNemar/binomial value is

$$p_{\text{McN}} = \min\{1, 2 \min(P_-, P_+)\}.$$

Concordant pairs do not provide evidence of a difference.

C.3 Effect Sizes and Confidence Intervals

The primary effect is

$$\hat{\Delta}_{sptr,a} = \frac{100}{Q} \sum_{q=1}^Q (y_{qspr} - y_{qspta}),$$

reported in percentage points. Here a is the comparator condition and all other indices are held fixed. We also report a continuity-corrected paired odds ratio

$$\widehat{\text{OR}}_{\text{paired}} = \frac{b + 0.5}{c + 0.5},$$

where b and c are the discordant counts defined above. Confidence intervals resample the $Q = 304$ whole-query cluster means with replacement 4,000 times using seed 20260630 and take the 2.5th and 97.5th bootstrap percentiles. Resampling individual rows would understate uncertainty because each query appears in many conditions.

C.4 BH False-Discovery Control

If M raw values are ordered $p_{(1)} \leq \dots \leq p_{(M)}$, the Benjamini-Hochberg procedure (Benjamini and Hochberg, 1995) computes the adjusted value

$$p_{(i)}^{\text{BH}} = \min_{j \geq i} \left\{ \min \left(1, \frac{M}{j} p_{(j)} \right) \right\}.$$

It controls the expected fraction of false discoveries among rejected hypotheses, not the probability of any false positive. The 252 mode-versus-direct cells form the primary family. Internal-reasoning configuration contrasts, target matching, best-CoT comparisons, persona moderation, and control-mechanism screens are separate scientific questions and therefore separate declared families.

C.5 Token Comparisons

Prompt, completion, and total-token differences are paired within query-persona-target, then averaged to one difference per query within each setting. For a token metric $z \in \{\text{prompt}, \text{completion}, \text{total}\}$, the query-level paired token difference is

$$d_{qsr,a}^z = \frac{1}{|\mathcal{P} \times \mathcal{T}|} \sum_{p,t} (z_{qspr} - z_{qspta}),$$

where $\mathcal{P}$ and $\mathcal{T}$ are the evaluated persona and target sets. Two-sided Wilcoxon signed-rank tests (Wilcoxon, 1945) operate on those 304 query-level differences, with BH applied separately to each token metric. Query-cluster bootstrap intervals target the mean token difference; the rank test targets a distributional location shift, so the two summaries need not agree for a heavy-tailed cell. Token use cannot establish whether a prefix was attended to or semantically ignored.

C.6 Clustered Logistic GEE

The complementary GEE contains 87,552 Bernoulli outcomes and 304 query clusters. GEE estimates population-averaged associations under correlated responses (Liang and Zeger, 1986). For observation i , let $q(i)$ denote its query cluster and let $\mu_i = E[y_i]$. We fit a logistic marginal mean model

$$\text{logit}(\mu_i) = \beta_0 + \beta^\top X_i,$$

where X_i contains model configuration, reasoning condition, persona, target, BIRD difficulty (Li et al., 2023), database, balanced ambiguity score, phrasing-specificity score, SCS, and the declared interaction terms. The estimating equation is

$$\sum_{q=1}^Q D_q^\top V_q^{-1} (Y_q - \mu_q) = 0,$$

where Y_q is the vector of binary outcomes for query q , μ_q is the corresponding fitted mean vector, $D_q = \partial \mu_q / \partial \beta$, and V_q is the working covariance matrix. The reported fit uses an independence working correlation and query-clustered robust sandwich covariance. The exchangeable working-correlation fit was numerically unstable and failed to converge; it was discarded. For an omnibus term represented by the contrast matrix R , the null hypothesis is $H_0 : R\beta = 0$ and the robust Wald statistic is

$$W = (R\hat{\beta})^\top \left[R \widehat{\text{Var}}(\hat{\beta}) R^\top \right]^{-1} (R\hat{\beta}),$$

where $\hat{\beta}$ is the fitted coefficient vector and $\widehat{\text{Var}}(\hat{\beta})$ is its query-clustered sandwich covariance; W is compared with a chi-square distribution having $\text{rank}(R)$ degrees of freedom. Table 8 lists the BH-adjusted omnibus Wald families.

C.7 Structural Complexity and Other Query Covariates

SCS is retained as a standardized query-level structural-complexity annotation in the adjusted GEE so that a reasoning-condition comparison is not mistaken for a difference in the structural demands of the query. Its BH-adjusted omnibus value in Table 8 is 0.362, so it does not provide a distinct adjusted association with correctness in

this panel. Difficulty is likewise non-significant after adjustment, whereas ambiguity and phrasing specificity remain associated with correctness. These variables are correlated features of the benchmark and persona rewrites, not independently manipulated causal treatments; the paired condition tests remain the primary evidence.

These procedures distinguish confirmatory paired comparisons from descriptive or focused analyses. The next section reports the Original- and Novice-specific families that were used to probe localized directions without replacing the primary family.

D Original and Novice Focused Families

Expert phrasing intentionally exposes more task structure, so we also analyzed Original and Novice phrasings as separate focused families. These analyses do not replace the full 252-test primary family. They ask whether less-explicit phrasing contains a clearer condition effect after removing expert prompts from the family under test. Table 9 gives the leading pooled focused contrasts.

Original phrasing contains positive raw movement, especially for SQL-side visible CoT, but no corrected discovery. Across the 48 Original-only explicit-CoT per-model cells, the smallest adjusted value is 0.481; across the 14 pooled Original all-condition tests, the smallest adjusted value is 0.196. Within the Original-only pooled contrasts, the largest positive SQL-side movements occur for pseudocode CoT, scrambled context, and NL CoT.

Novice phrasing gives the only focused corrected result, and it is not an explicit-CoT language: scrambled context improves SQL accuracy by 5.15 points after correction across the 14 Novice all-condition pooled tests. Explicit CoT remains non-significant for Novice phrasing; the smallest adjusted value in the pooled explicit-CoT family is 0.652. The narrowed GEE over Original and Novice rows converges and again does not make the omnibus mode family significant ($p_{\text{BH}} = 0.270$), while the mode-by-phrasing interaction is suggestive but not corrected ($p_{\text{BH}} = 0.109$).

An additional exploratory test averages the four explicit CoT assignments within query before comparing them with direct generation. Original-SQL is +1.56 points (95% query-bootstrap interval [0.23, 2.93], raw $p = 0.0239$), but does not cross the BH threshold across the four focused persona-target tests ($p_{\text{BH}} = 0.0954$). GLM4 has the clearest model-wise Original direction: +4.28 points for SQL (raw $p = 0.0154$, $p_{\text{BH}} = 0.0921$ across the six models in that persona-target family) and +3.13 points for Python (raw $p = 0.0355$). No aggregate explicit-CoT cell is a corrected discovery. Seventeen of 24 focused model-persona-target aggregate-CoT deltas are positive; Original-SQL has five positive configurations and one zero-delta configuration. This is a descriptive fixed-panel direction, not an independent-systems signifi-

cance test. These analyses were motivated after inspection and are labeled post hoc; they do not change the primary family.

D.1 Focused Control Mechanism Contrasts

The focused control analysis keeps Original and Novice phrasing separate and uses two matched comparators for every ablation. The NL-CoT comparison asks whether removing coherent task-relevant content, disrupting order, or introducing wrong-task content changes the visible-reasoning condition. The direct-generation comparison asks whether the same prefix changes accuracy at all relative to a prompt without visible CoT. Table 10 reports the corrected contrasts that localize the Novice-SQL scrambled result.

The Novice-SQL result is not a generic length effect: no corrected length-filler difference from direct is detected, while scrambled exceeds both filler and wrong content. Nor does it establish a benefit from incoherent reasoning. The interventions do not share one frozen trace, and the supplied prefix changes the model’s own generated reasoning behavior. The most specific supported claim is that the Novice-SQL configuration is sensitive to a scrambled reasoning-like context.

At the model level, scrambled Novice-SQL gains are +7.89 points for Gemini Flash-0, +7.24 for GLM4, and +6.58 for Qwen3. These three survive correction over the 18 controls inside the Novice-SQL screen, but none survives the wider 72-cell focused-control correction. The pooled fixed-panel result therefore captures a repeated directional effect within the completed panel.

The difficulty split is shown separately in Table 11. Novice scrambled contrasts are positive across all three difficulty levels, with the strongest corrected evidence on simple and challenging queries and a smaller direct-generation contrast on moderate queries. The corresponding Original contrasts do not survive their correction family. This localization is consistent with a phrasing-dependent sensitivity, not a difficulty-invariant property of token permutation.

The focused control results motivate a separate look at internal-reasoning configuration, which is the next channel of variation examined below.

E Internal-Reasoning Configuration Detail

The 96-contrast family contains two model families, three personas, two targets, and eight reasoning conditions. Sixteen cells survive BH. Gemma4 accounts for 15: SQL-CoT under Expert-Python; five Novice-Python visible/control cells; all eight Original-Python conditions, including direct; and wrong-content Original-SQL. Gemini Flash contributes one corrected cell: wrong-content Original-SQL,

where Flash-24k is 7.89 points below Flash-0. This distribution explains why a single pooled “internal reasoning helps” coefficient would be misleading. Figure 4 shows the complete configuration-contrast matrix.

The contrast matrix establishes that changing the internal-reasoning configuration is not a single global factor. It does not identify the contents or amount of model-side internal reasoning: the analysis observes only accuracy, resources, and the externally visible response trace under each configuration. The next section evaluates whether target-matched explicit CoT or a selected best format provides a corresponding global rule.

F Target Matching, Controls, and Best Formats

No target-matched comparison survives BH across 36 cells, and no selected best CoT is significantly superior to every alternative in its corrected family. This is not evidence that representations are interchangeable; interaction tests show that their effects vary by persona and model. It is evidence against a single global target-matching rule.

Six cells survive the narrower seven-tests-within-cell correction, but none survives the 252-test primary family. Three are scrambled Novice-SQL gains (GLM4, Qwen3, and Gemini Flash-0); two are GLM4 Expert-Python losses for NL/SQL CoT; one is a Gemma4-on Expert-Python SQL-CoT gain. These are localized within-cell results, not corrected discoveries in the primary family.

Because a representation can change resource use even when it has no global accuracy advantage, the next section characterizes the associated token and latency trade-offs.

G Efficiency Detail

The efficiency analysis treats each matched model-persona-target-condition cell as an accuracy-resource point. We report direct generation and each of the four explicit CoTs separately; controls remain in the ablation and step-compliance analyses but are not assigned a cost/latency comparison here. This makes the marginal cost of each visible CoT explicit without pretending that a single accuracy-per-token ratio is comparable across providers. Table 12 reports the individual formats, while Table 13 gives their compact mean and Table 14 separates task difficulty from prefix overhead.

The bucket tables use model-adaptive bins. The initial width targets roughly 10% of a model’s observed resource span, is rounded to a readable standard width, and is widened when necessary to retain at most ten plotted bucket positions in a model panel. Every explicit CoT format remains separate. Figures 5 and 6 use persona-faceted scatter plots with the six model configurations

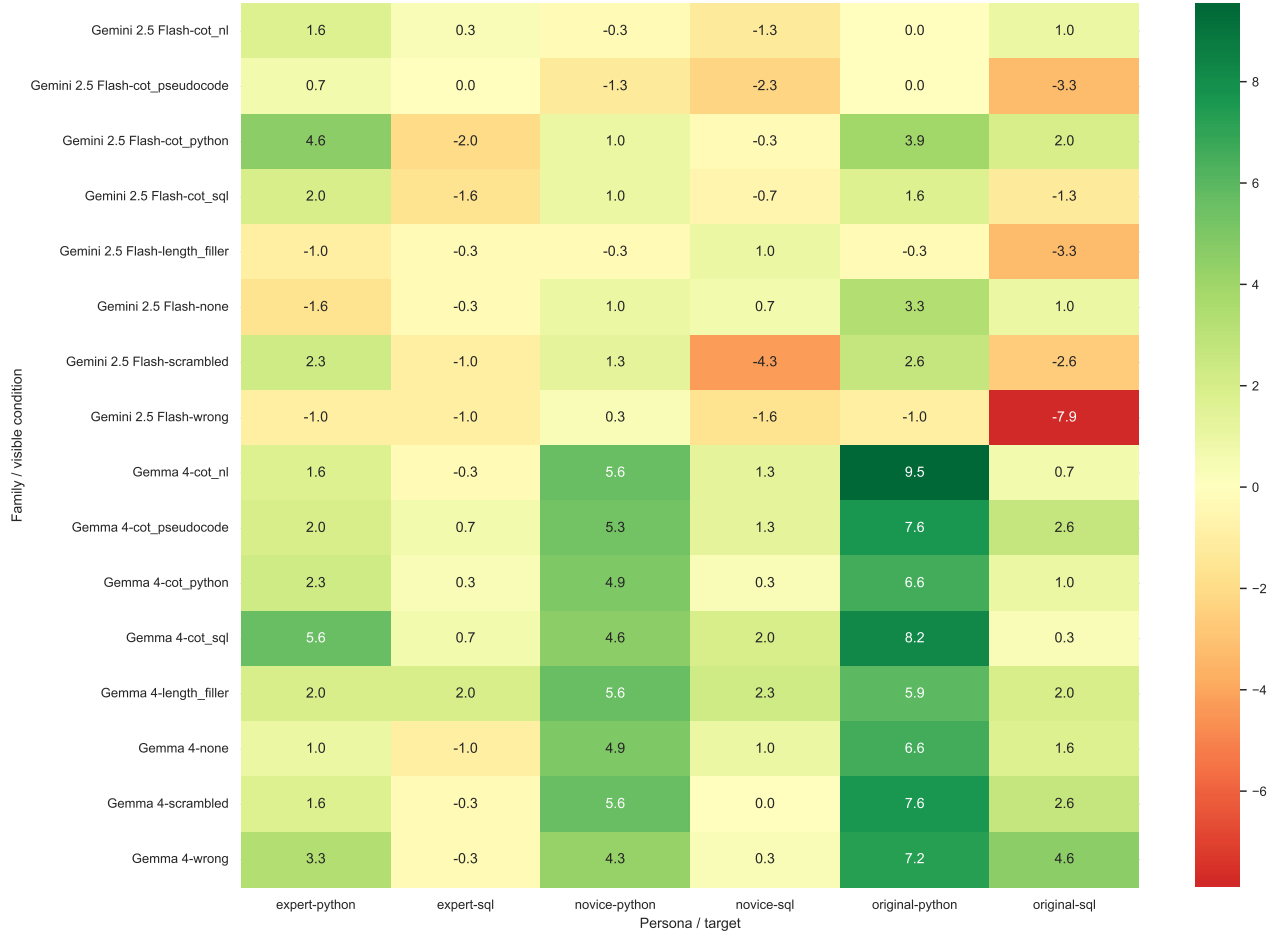


Figure 4: Complete same-family internal-reasoning effects (enabled minus disabled, or budget 24,576 minus 0).

arranged in a 2-by-3 layout inside each Original, Novice, and Expert block. Color encodes the reasoning condition, marker shape identifies the target language, and point size encodes difficulty; each point is the mean accuracy within the corresponding model-adaptive bucket. The y-axis is tightened separately within each persona row block so that the central mass of points is easier to read. Bucket widths target roughly 10% of the observed span, are rounded to readable standard widths, and are widened for high-range panels when needed to avoid more than ten plotted positions. Harder tasks move both direct and CoT conditions toward larger resource budgets, while each explicit CoT adds further overhead whose accuracy return is model- and target-dependent. Figure 7 applies the same signed-bucket construction to enabled-minus-disabled thinking changes for each model family.

Table 15 lists representative accuracy, token, and latency points used to interpret cost trade-offs.

Every control increases both completion and total tokens relative to direct generation in all six configurations after BH correction. Compared with the mean of the four explicit-CoT conditions, control completions are shorter in 14 of 18 model-by-control cells, and all 14 differences are corrected. Shorter output does not identify

noise rejection: controls still generate 8.6-10.2 thinking steps on average. Exact adherence to the intended eight-step count is 58.5% for length filler, 15.6% for scrambled, and 39.2% for wrong content, compared with 99.5-99.9% for explicit CoT. The control signature is therefore shorter but more fragmented generated reasoning plus a positive token overhead relative to direct generation. Figure 8 shows the exact-step compliance pattern behind this interpretation. Table 16 then tests whether exact-step compliance is associated with correctness within each control after adjustment for model configuration, persona, target, and difficulty.

Compliance is positively associated with correctness for length filler and wrong content, but not for scrambled context (Table 16). This difference supports two behavioral axes: adherence to the visible output structure for the labeled controls, and a separate lexical/context sensitivity after token permutation. Because compliance is observed rather than randomized, these adjusted associations do not establish that producing exactly eight steps causes a correct answer.

The bucket figures should be interpreted as budget-stratified descriptive comparisons, not causal dose-response curves. Formal equivalence or non-inferiority

tests with predeclared margins are needed before recommending a cheaper configuration as a substitute.

References

- Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(1):289-300, 1995. doi: <https://doi.org/10.1111/j.2517-6161.1995.tb02031.x>. URL <https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1995.tb02031.x>.
- Saumya Chaturvedi, Aman Chadha, and Laurent Bind-schaedler. SQL-of-thought: Multi-agent text-to-SQL with guided error correction. In *NeurIPS 2025 Fourth Workshop on Deep Learning for Code*, 2025. URL <https://openreview.net/forum?id=NBDa9012jE>.
- Xun Deng, Sicheng Zhong, Honghua Dong, Jingyu Hu, Sidi Mohamed Beillahi, Xujie Si, and Fan Long. Assessing code generation with intermediate languages. *arXiv preprint arXiv:2407.05411*, 2024. URL <https://arxiv.org/abs/2407.05411>.
- Zhongjun Ding, Yin Lin, Tianjing Zeng, Rong Zhu, Bolin Ding, and Jingren Zhou. Ambisql: Interactive ambiguity detection and resolution for text-to-sql. In *Companion of the International Conference on Management of Data*, SIGMOD Companion '26, page 26–29, New York, NY, USA, 2026. Association for Computing Machinery. ISBN 9798400724503. doi: 10.1145/3788853.3801581. URL <https://doi.org/10.1145/3788853.3801581>.
- Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 10764-10799, 23-29 Jul 2023. URL <https://proceedings.mlr.press/v202/gao23f.html>.
- Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *Second Conference on Language Modeling*, 2025. URL <https://openreview.net/forum?id=Itxz7S4Ip3>.
- Mingqian He, Yongliang Shen, Wenqi Zhang, Qiuying Peng, Jun Wang, and Weiming Lu. STaR-SQL: Self-taught reasoner for text-to-SQL. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 24365-24375, Vienna, Austria, July 2025. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1187. URL <https://aclanthology.org/2025.acl-long.1187/>.
- Davin Hill, Pawan Chorasaya, Bhawani Shankar Leelar, Oren Avram, Robert E. Tillman, and Tamer Soliman. Bird-persona: A persona-aware benchmark for sql and pandas analytics code generation. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing (EMNLP 2026): Industry Track*, Budapest, Hungary, October 2026. Association for Computational Linguistics.
- Hangle Hu, Chenyu Hou, Bin Cao, and Ruizhe Li. Benchmarking text-to-python against text-to-sql: The impact of explicit logic and ambiguity. *arXiv preprint arXiv:2601.15728*, 2026. URL <https://arxiv.org/abs/2601.15728>.
- Naizhu Jin, Zhong Li, Guang Yang, Tian Zhang, and Qingkai Zeng. Understanding chain-of-thought effectiveness in code generation: An empirical and information-theoretic analysis. 2025a. URL <https://arxiv.org/abs/2512.09679>.
- Naizhu Jin, Zhong Li, Tian Zhang, and Qingkai Zeng. MSCoT: Structured chain-of-thought generation for multiple programming languages. In *2025 International Joint Conference on Neural Networks (IJCNN)*, pages 1-8, 2025b. doi: 10.1109/IJCNN64981.2025.11228673.
- Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Wen-Tau Yih, Daniel Fried, Sida Wang, and Tao Yu. DS-1000: A natural and reliable benchmark for data science code generation. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 18319-18345, 23-29 Jul 2023. URL <https://proceedings.mlr.press/v202/lai23b.html>.
- Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023. URL <https://arxiv.org/abs/2307.13702>.
- Jinyang Li, Binyuan Hui, Ge Qu, Jiayi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin Chang, Fei Huang, Reynold Cheng, and Yongbin Li. Can LLM already serve as a database interface? a BIG bench for large-scale database grounded text-to-SQLs. In *Advances in Neural Information Processing Systems*, volume 36, pages 42330-42357, 2023. doi: 10.52202/075280-1835. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/83fc8fab1710363050bbd1d4b8cc0021-Paper-Datasets_and_Benchmarks.pdf.
- Kung-Yee Liang and Scott L. Zeger. Longitudinal data analysis using generalized linear models. *Biometrika*, 73(1):13-22, 1986. ISSN 00063444. URL <http://www.jstor.org/stable/2336267>.

- Limin Ma, Ken Pu, Ying Zhu, and Wesley Taylor. Comparing large language models for generating complex queries. *Journal of Computer and Communications*, 13(2):236-249, 2025.
- Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153-157, 1947. doi: 10.1007/BF02295996.
- Shuyin Ouyang, Dong Huang, Jingwen Guo, Zeyu Sun, Qihao Zhu, and Jie M. Zhang. DSCodebench: A realistic benchmark for data science code generation. 2025. URL <https://arxiv.org/abs/2505.15621>.
- Haritz Puerto, Martin Tutek, Somak Aditya, Xiaodan Zhu, and Iryna Gurevych. Code prompting elicits conditional reasoning abilities in Text+Code LLMs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11234-11258, Miami, Florida, USA, November 2024. doi: 10.18653/v1/2024.emnlp-main.629. URL <https://aclanthology.org/2024.emnlp-main.629/>.
- Charlie Victor Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=4FWAwZtd2n>.
- Ian Su, Gaurav Purushothaman, Jey Narayan, Ruhika Goel, Kevin Zhu, Sunishchal Dev, Yash More, and Maheep Chaudhary. Broken chains: The cost of incomplete reasoning in llms. *arXiv preprint arXiv:2602.14444*, 2026. URL <https://arxiv.org/abs/2602.14444>.
- Tao Sun, Linzheng Chai, Jian Yang, Yuwei Yin, Hongcheng Guo, Jiaheng Liu, Bing Wang, Liqun Yang, and Zhoujun Li. UniCoder: Scaling code large language model via universal code. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1812-1824, Bangkok, Thailand, August 2024. doi: 10.18653/v1/2024.acl-long.100. URL <https://aclanthology.org/2024.acl-long.100/>.
- Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems*, volume 36, pages 74952-74965, 2023. doi: 10.52202/075280-3275. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/ed3fea9033a80fea1376299fa7863f4a-Paper-Conference.pdf.
- Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. Towards understanding chain-of-thought prompting: An empirical study of what matters. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2717-2739, Toronto, Canada, July 2023. doi: 10.18653/v1/2023.acl-long.153. URL <https://aclanthology.org/2023.acl-long.153/>.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, pages 24824-24837, 2022. doi: 10.52202/068431-1800. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.
- Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80-83, 1945. doi: 10.2307/3001968. URL <https://doi.org/10.2307/3001968>.
- Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911-3921, Brussels, Belgium, October-November 2018. doi: 10.18653/v1/D18-1425. URL <https://aclanthology.org/D18-1425/>.
- Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D. Goodman. Quiet-STaR: Language models can teach themselves to think before speaking. *arXiv preprint arXiv:2403.09629*, 2024. URL <https://arxiv.org/abs/2403.09629>.
- Tianshi Zheng, Yixiang Chen, Chengxi Li, Chunyang Li, Qing Zong, Haochen Shi, Baixuan Xu, Yangqiu Song, Ginny Wong, and Simon See. The curse of CoT: On the limitations of chain-of-thought in in-context learning. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL <https://openreview.net/forum?id=7SIrcvYNYj>.
- Zhanke Zhou, Rong Tao, Jianing Zhu, Yiwen Luo, Zengmao Wang, and Bo Han. Can language models perform robust reasoning in chain-of-thought prompting with noisy rationales? In *Advances in Neural Information Processing Systems*, volume 37, pages 123846-123910, 2024. doi: 10.52202/079017-3936. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/dfaa29ed28dfa175bcc5e2a54aa199f8-Paper-Conference.pdf.
- Yuqi Zhu, Ge Li, Xue Jiang, Jia Li, Hong Mei, Zhi Jin, and Yihong Dong. Uncertainty-guided chain-of-thought for code generation with llms. *CoRR*,

abs/2503.15341, 2025. doi: 10.48550/ARXIV.2503.
15341. URL [https://doi.org/10.48550/arXiv.2503.
15341](https://doi.org/10.48550/arXiv.2503.15341).

Model	Persona	Target	Direct	NL CoT	SQL CoT	Python CoT	Pseudo code CoT	Length filler	Scrambled	Wrong content
GLM4	expert	SQL	88.16	87.83	86.18	87.83	88.82	90.79	86.84	87.83
GLM4	expert	PYTHON	60.20	54.61	53.95	55.59	57.89	60.20	60.86	55.92
GLM4	novice	SQL	19.08	21.38	18.42	21.05	18.42	20.72	26.32	20.72
GLM4	novice	PYTHON	14.14	14.80	13.49	15.13	15.13	14.14	16.45	13.16
GLM4	original	SQL	23.68	28.29	28.29	25.99	29.28	24.01	25.66	27.30
GLM4	original	PYTHON	15.13	19.74	18.42	17.11	17.76	16.45	15.13	15.79
Gemini Flash-0	expert	SQL	92.43	91.45	93.42	93.42	90.79	92.11	92.11	92.11
Gemini Flash-0	expert	PYTHON	68.42	65.13	67.11	64.14	67.76	68.42	66.45	66.78
Gemini Flash-0	novice	SQL	30.92	32.57	33.55	30.92	33.22	30.92	38.82	34.21
Gemini Flash-0	novice	PYTHON	20.39	22.70	23.36	20.72	22.70	21.38	24.01	19.41
Gemini Flash-0	original	SQL	35.86	36.51	37.17	35.53	39.80	39.47	39.14	41.12
Gemini Flash-0	original	PYTHON	23.03	25.00	25.99	22.04	26.32	26.97	25.00	24.34
Gemini Flash-24k	expert	SQL	92.11	91.78	91.78	91.45	90.79	91.78	91.12	91.12
Gemini Flash-24k	expert	PYTHON	66.78	66.78	69.08	68.75	68.42	67.43	68.75	65.79
Gemini Flash-24k	novice	SQL	31.58	31.25	32.89	30.59	30.92	31.91	34.54	32.57
Gemini Flash-24k	novice	PYTHON	21.38	22.37	24.34	21.71	21.38	21.05	25.33	19.74
Gemini Flash-24k	original	SQL	36.84	37.50	35.86	37.50	36.51	36.18	36.51	33.22
Gemini Flash-24k	original	PYTHON	26.32	25.00	27.63	25.99	26.32	26.64	27.63	23.36
Gemma4-off	expert	SQL	93.75	93.09	93.75	92.43	93.42	92.43	93.75	93.75
Gemma4-off	expert	PYTHON	64.80	64.80	64.47	64.80	66.12	64.14	64.80	64.47
Gemma4-off	novice	SQL	29.61	30.26	28.95	30.59	29.93	27.63	33.22	30.92
Gemma4-off	novice	PYTHON	18.09	17.11	17.11	17.43	16.78	15.79	18.42	17.43
Gemma4-off	original	SQL	36.18	37.17	37.83	37.50	36.51	37.17	36.84	36.51
Gemma4-off	original	PYTHON	19.74	17.76	19.41	19.74	18.09	18.42	18.42	18.75
Gemma4-on	expert	SQL	92.76	92.76	94.41	92.76	94.08	94.41	93.42	93.42
Gemma4-on	expert	PYTHON	65.79	66.45	70.07	67.11	68.09	66.12	66.45	67.76
Gemma4-on	novice	SQL	30.59	31.58	30.92	30.92	31.25	29.93	33.22	31.25
Gemma4-on	novice	PYTHON	23.03	22.70	21.71	22.37	22.04	21.38	24.01	21.71
Gemma4-on	original	SQL	37.83	37.83	38.16	38.49	39.14	39.14	39.47	41.12
Gemma4-on	original	PYTHON	26.32	27.30	27.63	26.32	25.66	24.34	25.99	25.99
Qwen3	expert	SQL	91.12	89.80	90.79	89.47	90.79	91.78	92.43	91.12
Qwen3	expert	PYTHON	58.88	58.88	59.87	54.61	58.22	61.84	62.83	62.17
Qwen3	novice	SQL	23.03	25.99	23.03	23.68	25.33	24.34	29.61	25.33
Qwen3	novice	PYTHON	16.12	19.08	16.12	13.82	17.43	18.09	18.42	19.74
Qwen3	original	SQL	29.93	33.55	30.26	33.55	30.59	31.58	34.54	30.26
Qwen3	original	PYTHON	21.71	22.04	21.38	20.39	20.72	23.03	25.33	24.01

Table 7: Full persona-wise exact execution accuracy (%). The best condition in each model-persona-target row is bold.

Accuracy within model-adaptive tokens bins: Original, Novice, Expert phrasing

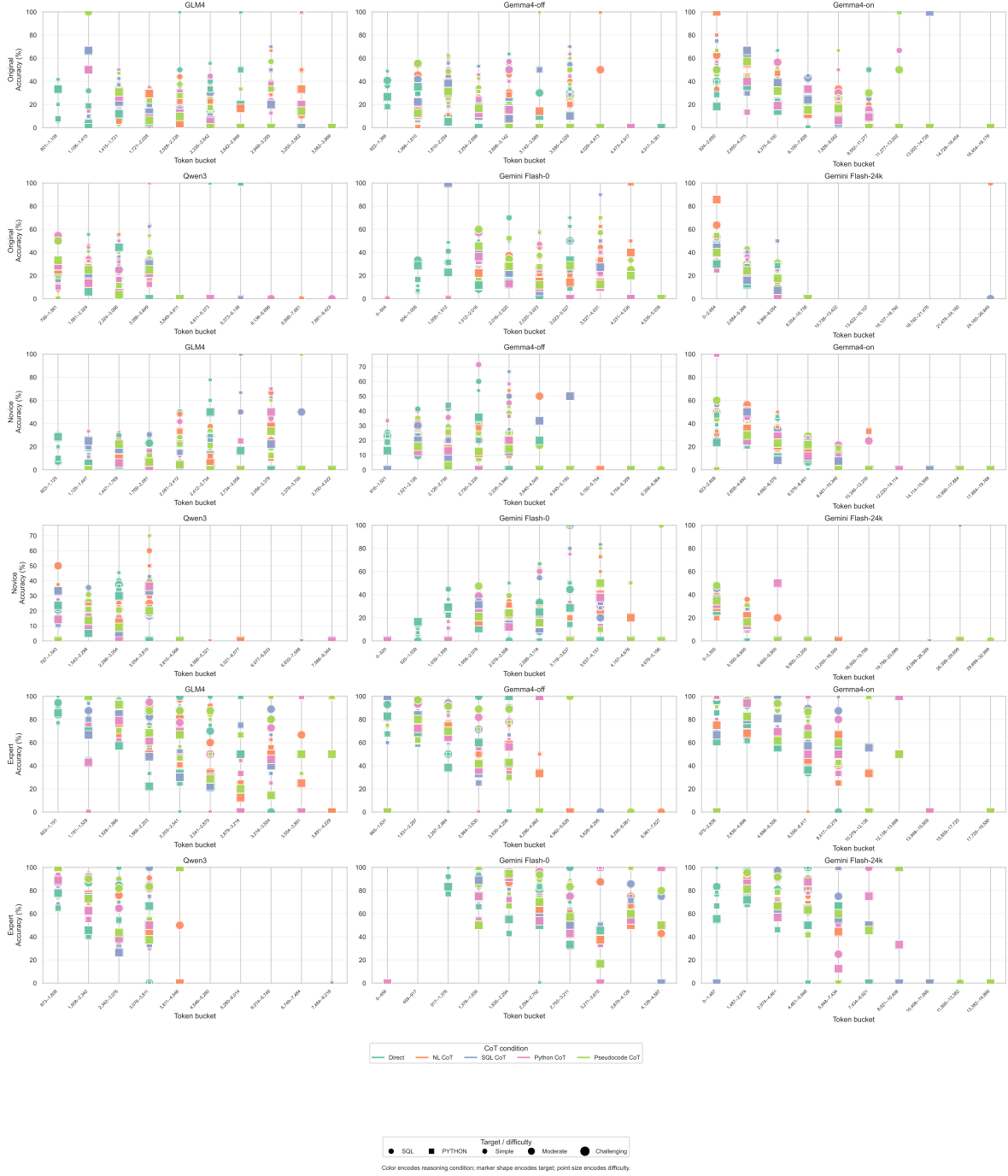


Figure 5: Accuracy within model-adaptive token bins, stratified by model, persona, target, difficulty, and reasoning condition. Original, Novice, and Expert are shown; point size encodes difficulty and the y-axis is tightened within each persona row block. Bucket widths target roughly 10% of the span and widen when necessary to keep high-range panels readable.

Accuracy within model-adaptive latency bins: Original, Novice, Expert phrasing

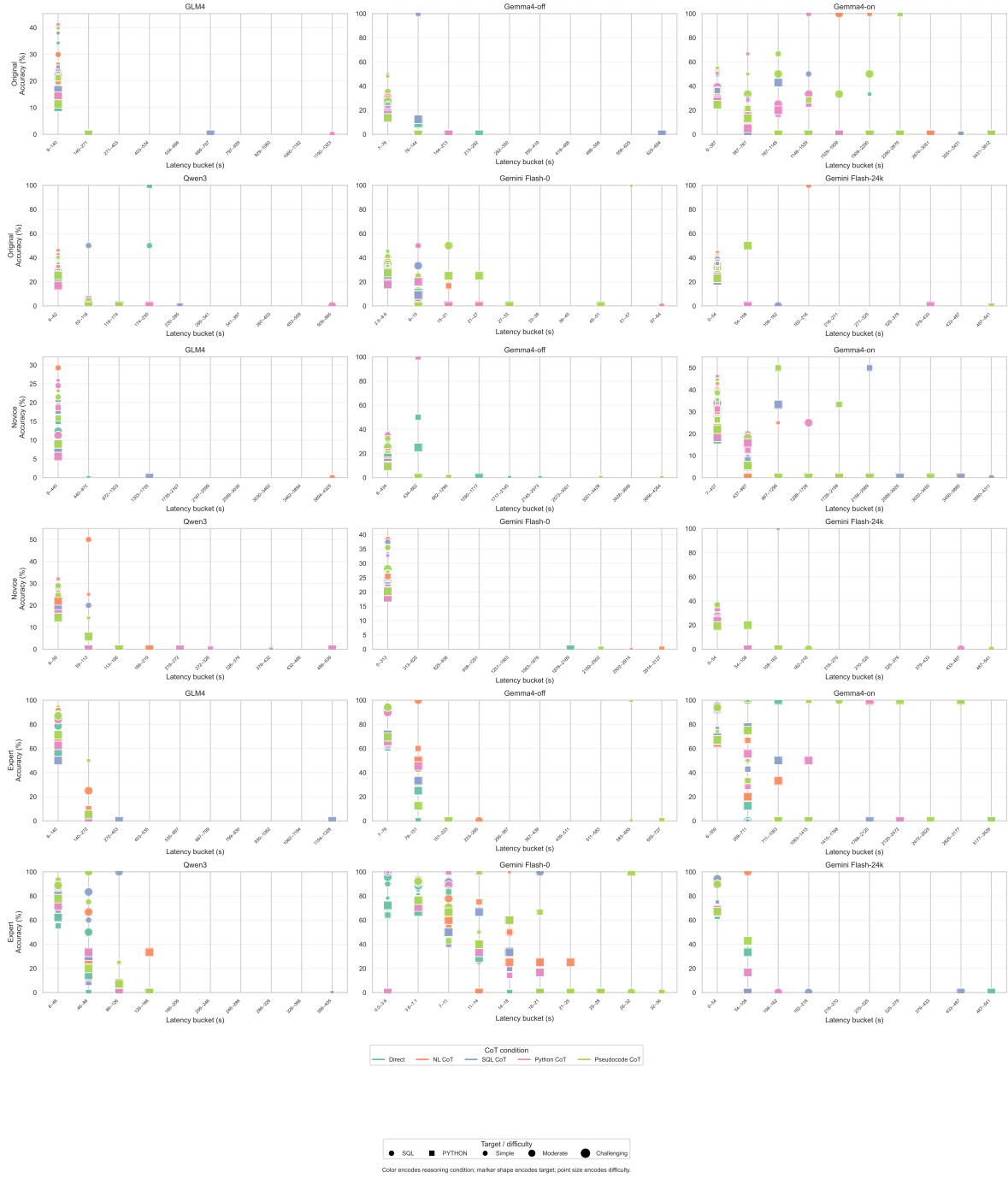


Figure 6: Accuracy within model-adaptive latency bins, stratified by model, persona, target, difficulty, and reasoning condition. Original, Novice, and Expert are shown; point size encodes difficulty and the y-axis is tightened within each persona row block. Bucket widths are chosen from roughly 10% of the observed span and rounded to the nearest standard latency bin.

Thinking-toggle accuracy changes by signed resource bucket: Original, Novice, Expert phrasing

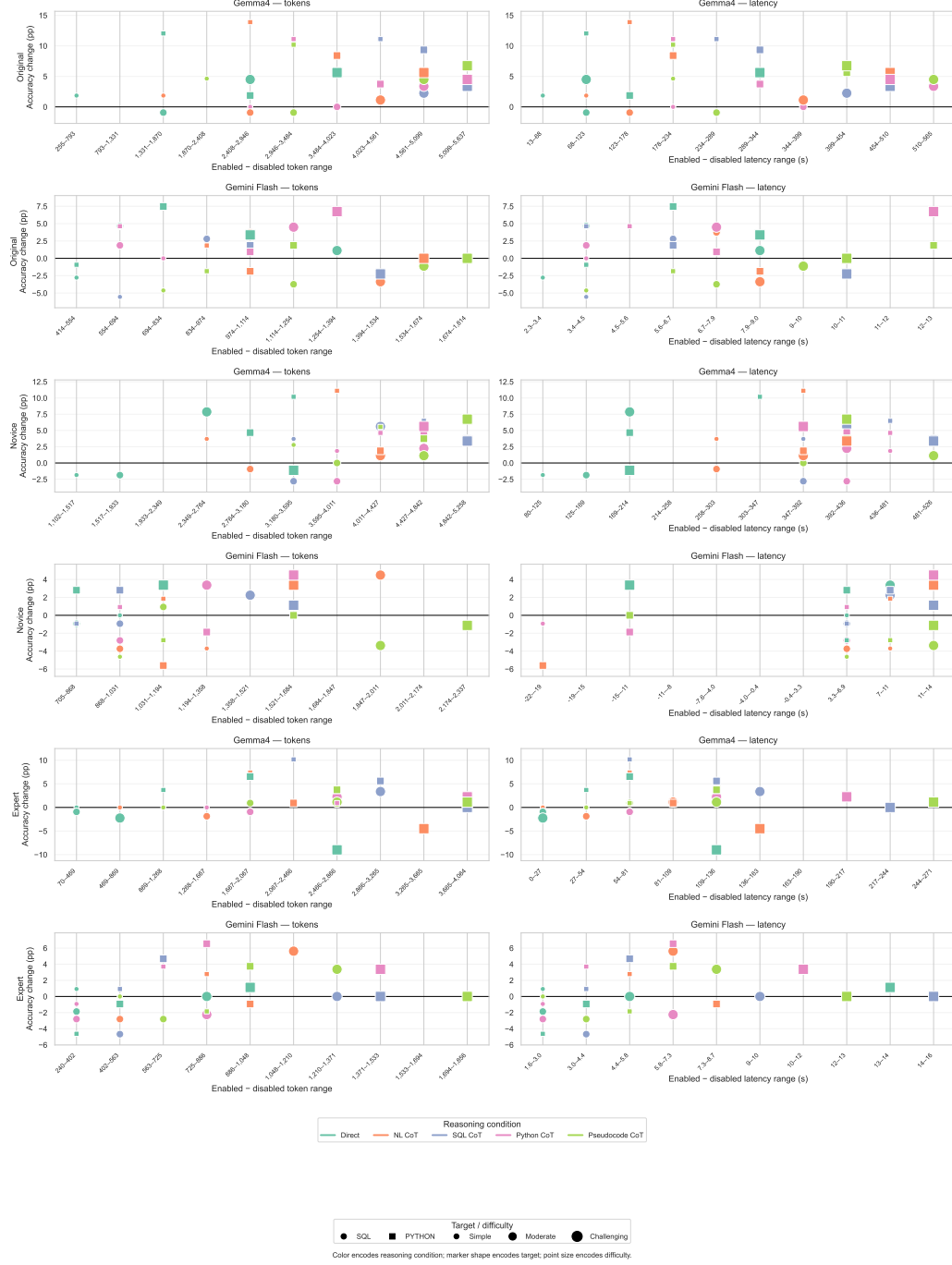


Figure 7: Internal-reasoning configuration effects in signed token and latency buckets for Original, Novice, and Expert phrasing. Color encodes the reasoning condition, marker shape target, and marker size difficulty.

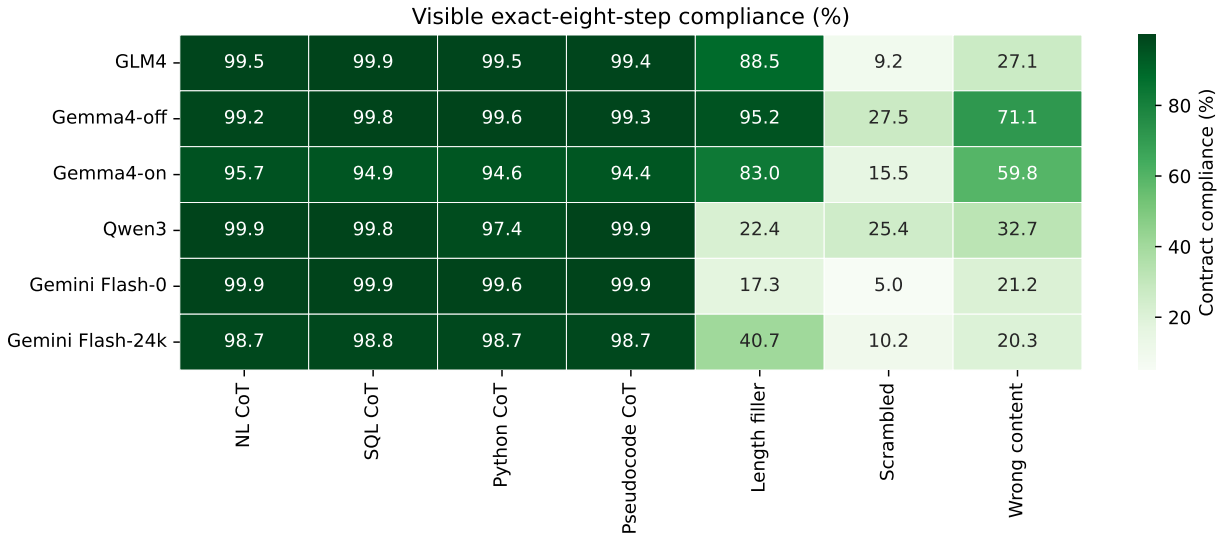


Figure 8: Exact eight-step contract compliance for explicit-CoT and control conditions by model. Low compliance, especially under scrambled context, argues against interpreting lower completion length as evidence that the control was simply ignored.

Term family	p_{BH}
Persona	$< 10^{-6}$
Target	$< 10^{-26}$
Model configuration	$< 10^{-11}$
Mode	0.620
Mode $\times$ persona	0.00033
Mode $\times$ target	0.626
Model configuration $\times$ mode	0.0030
Difficulty	0.179
SCS	0.362
Ambiguity	0.0016
Phrasing specificity	0.0017

Table 8: Complementary robust-GEE omnibus tests.
Paired exact contrasts remain the primary inference.

Phrasing	Target	Condition	Direct	Condition	Δ pp	p_{BH}
Original	SQL	Pseudocode CoT	33.39	35.31	1.92	0.168
Original	SQL	NL CoT	33.39	35.14	1.75	0.168
Original	SQL	Scrambled	33.39	35.36	1.97	0.196
Novice	SQL	Scrambled	27.47	32.62	5.15	0.00052
Novice	Python	Scrambled	18.86	21.11	2.25	0.111
Novice	SQL	NL CoT	27.47	28.84	1.37	0.439

Table 9: Top pooled condition-versus-direct results in the focused Original and Novice families. Pooled contrasts average paired deltas across the six configurations within each query before testing.

Phrasing	Target	Contrast	Δ pp	95% CI	Raw p	p_{BH}
Novice	SQL	Scrambled – direct	5.15	[2.80, 7.62]	3.72×10^{-5}	0.00045
Novice	SQL	Scrambled – length filler	5.04	[3.07, 7.07]	9.14×10^{-7}	0.000011
Novice	SQL	Scrambled – wrong content	3.45	[1.54, 5.48]	0.00063	0.0038
Novice	SQL	Scrambled – explicit-CoT mean	4.39	[2.43, 6.41]	1.61×10^{-5}	0.00019
Novice	Python	Scrambled – length filler	2.47	[0.77, 4.22]	0.0046	0.0184
Novice	Python	Scrambled – wrong content	2.58	[0.77, 4.50]	0.0083	0.0250

Table 10: Corrected focused control contrasts. BH is applied within three separate 12-test families: controls versus direct, between controls, and controls versus the four-format explicit-CoT mean.

Persona	Difficulty	Δ vs NL CoT	q_{BH}	Δ vs direct	q_{BH}
Original	Simple	0.00	1.0000	1.85	0.1646
Original	Moderate	-0.62	0.8076	0.23	0.9469
Original	Challenging	1.31	0.4539	2.34	0.1646
Novice	Simple	2.85	0.0183	4.01	0.0026
Novice	Moderate	1.17	0.4539	3.04	0.0183
Novice	Challenging	3.84	0.0077	4.12	0.0034

Table 11: Scrambled-minus-comparator accuracy contrasts by persona and difficulty, pooled across the six configurations and both targets. Deltas are percentage points; q_{BH} values use the focused difficulty family.

Reasoning condition	Accuracy (%)	Mean tokens	Mean latency (s)	Δ pp	Extra tokens	Extra latency (s)
Direct	42.94	2,153	37.1	0.00	0	0.0
NL CoT	43.41	3,095	69.3	0.48	942	32.2
SQL CoT	43.52	3,077	75.4	0.58	924	38.4
Python CoT	42.82	3,069	76.0	-0.12	916	38.9
Pseudocode CoT	43.51	3,194	81.3	0.58	1,041	44.2

Table 12: Pooled cost and accuracy for each explicit CoT format versus direct generation over the six configurations, three personas, and two targets. Bold marks the highest accuracy and the smallest total-token and latency values.

Condition family	Accuracy (%)	Tokens	Latency (s)	Δ pp	Extra tokens	Extra latency (s)	Dominated (T/L, %)
Direct	42.94	2,153	37.1	0.00	0	0.0	0.0 / 0.0
Explicit CoT mean	43.32	3,109	75.5	0.38	956	38.4	41.7 / 38.9

Table 13: Pooled descriptive efficiency summary over six configurations, three personas, and two targets. Direct-dominated percentages count matched cells where direct generation is at least as accurate and no more costly, with one strict advantage.

Difficulty	Condition family	Accuracy (%)	Mean tokens	Mean latency (s)	Δ pp vs direct
Simple	Direct	47.25	1,943	31.9	0.00
Simple	Explicit CoT mean	47.69	2,834	61.8	0.44
Moderate	Direct	42.52	2,128	34.2	0.00
Moderate	Explicit CoT mean	43.12	3,079	73.1	0.60
Challenging	Direct	38.20	2,439	46.9	0.00
Challenging	Explicit CoT mean	38.25	3,478	95.0	0.05

Table 14: Difficulty-stratified efficiency means pooled across model, persona, and target cells. Difficulty raises both baseline and prompted resource use; the table should not be interpreted as a causal difficulty effect. From simple to challenging, direct generation adds approximately 496 tokens and 15.0 seconds, while explicit CoT adds 644 tokens and 33.2 seconds.

Model / CoT	Accuracy	Tokens	Latency (s)
GLM4 direct	36.73	1,537	20.5
GLM4 NL CoT	37.77	2,222	63.8
Qwen3 direct	40.13	1,549	13.6
Gemma4-off direct	43.70	1,873	28.8
Gemma4-on direct	46.05	3,743	144.7
Gemma4-on SQL CoT	47.15	6,056	318.3
Gemini Flash-0 direct	45.18	1,742	5.4
Gemini Flash-24k direct	45.83	2,475	9.5

Table 15: Selected pooled efficiency points. Accuracy is percent; latency is locally measured client-observed wall-clock time and is not hardware-normalized.

Control	Correctness OR	q_{BH}
Length filler	1.325	0.00048
Scrambled	0.995	0.94542
Wrong content	1.263	0.00264

Table 16: Adjusted association between exact eight-step compliance and correctness within each control. Odds ratios are from query-clustered logistic GEE; BH correction is across the three control-specific tests.