DATA2DECISION: A PRESCRIPTIVE ANALYTICS DATA AGENT BRIDGING ENTERPRISE INFORMATION AND OPTIMAL DECISIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Enterprise business analytics has evolved from simple reporting to sophisticated decision-support systems. Prescriptive analytics, as the most advanced form of business analytics, aims to recommend optimal actions based on data, yet the field lacks standardized benchmarks that reflect real-world complexity where optimization parameters must be extracted from enterprise databases. We present `Data2Decision`, the first data agent specifically designed for database-grounded prescriptive analytics that produces mathematically optimal decisions, accompanied by `Schema2Opt`, a comprehensive benchmark simulating enterprise decision environments. `Schema2Opt` transforms SQL database schemas into realistic optimization problems, providing complete business contexts, operational databases, and verified solutions. Our `Data2Decision` agent tackles these scenarios through a two-stage pipeline with test-time scaling: first extracting optimization parameters from databases via SQL generation, then formulating and solving optimization models with multi-solver consensus, achieving end-to-end automation without manual preprocessing.

1 INTRODUCTION

Enterprise decision-making has undergone a fundamental transformation in the digital age (Islam, 2024; Kraus et al., 2021). As organizations generate unprecedented volumes of data, business leaders increasingly rely on data-driven approaches rather than intuition-based decision-making (McAfee et al., 2012; Davenport & Harris, 2017). This shift has driven the evolution of enterprise business analytics from simple reporting systems to sophisticated decision-support frameworks that can directly impact organizational performance and competitive advantage. However, most existing business analytics methods focus primarily on *descriptive analytics* (Hong et al., 2024; Katsogiannis-Meimarakis & Koutrika, 2023; Islam, 2024), understanding past data characteristics. Text-to-SQL systems, while powerful for data retrieval and exploration (Zhong et al., 2017; Yu et al., 2018), are fundamentally descriptive analytics tools designed to answer historical questions rather than guide future actions. In contrast, *prescriptive analytics* represents the most advanced form of business analytics, recommending optimal actions through mathematical optimization and decision modeling (Lepenioti et al., 2020; Wissuchek & Zschech, 2024). This inherent limitation of Text-to-SQL and other descriptive approaches creates a significant gap between data analysis and actionable decision-making in enterprise environments, particularly when decisions require mathematical optimization to identify provably optimal solutions.

The emerging field of Text-to-OPT has made mathematical modeling more accessible by enabling natural language interfaces to optimization solvers (Tang et al., 2023; Wang et al., 2024; Zheng et al., 2024). Despite recent advances, current approaches suffer from a critical limitation: they assume optimization parameters are explicitly provided within problem descriptions as LaTeX tables or textbook-style scenarios (AhmadiTeshnizi et al., 2024; Zhang & Luo, 2025). This oversimplification misaligns with real-world enterprise workflows (Figure 1), where OR experts must collaborate with business managers and data engineers to extract decision-relevant information from complex operational databases. While some recent work has explored prescriptive analytics tasks and data extraction from database tables, such as `InsightBench` (Sahu et al., 2025), these approaches typ-

ically provide qualitative, single-dimensional action recommendations (e.g., open an incident ticket) rather than quantitative, multi-variable optimization solutions with verifiable objective values.

Beyond methodological oversimplification, existing Text-to-OPT benchmarking datasets suffer from a data generation paradigm issue. Current datasets follow a *top-down* approach: they start with textbook optimization problems (e.g., TSP, knapsack, facility location) and create variations by adjusting parameters (Xiao et al., 2023; Yang et al., 2025; Lu et al., 2025). This paradigm confines the field to recycling known formulations rather than discovering what optimization opportunities actually exist in enterprise data. Critically, this limitation constrains the potential of even the most advanced Text-to-OPT methods. Fine-tuning approaches (Tang et al., 2023; Jiang et al., 2024), as well as recent reinforcement learning methods with verifiable rewards such as SIRL (Chen et al., 2025), are bounded by this paradigm. These methods can only learn to replicate existing human optimization knowledge rather than discover novel optimization opportunities from real-world data. This raises a crucial question: *can we invert this process by starting from existing database schemas to discover latent optimization opportunities that may not exist in operations research literature?*

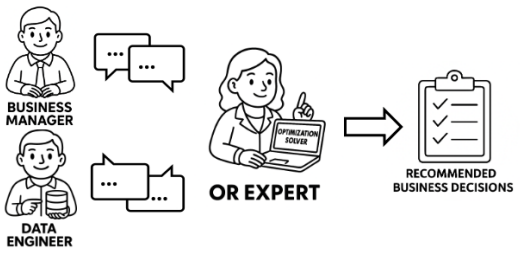

Figure 1: Real-world enterprise decision-making involves collaborative workflows where operation research (OR) experts work with business managers and data engineers to produce optimal business decisions with solvers.

To bridge this gap, we present `Schema2Opt`, the first benchmark dataset designed with a *bottom-up* philosophy that transforms real SQL database schemas into realistic optimization problems. Unlike existing benchmarks, our dataset generation pipeline discovers optimization opportunities directly from database structures. Based on this benchmark, we propose `Data2Decision`, the first data agent specifically designed for database-grounded prescriptive analytics that produces mathematically optimal decisions. Specifically, we make the following contributions:

- **Prescriptive Analytics Problem Definition**: We propose a realistic prescriptive analytics problem formulation that goes beyond simple descriptive analysis based on Text-to-SQL systems and addresses the practical challenges of real-world enterprise decision-making. Unlike existing Text-to-OPT approaches that assume pre-embedded parameters, our formulation requires extracting optimization parameters from enterprise databases, reflecting actual business analytics workflows.
- **Novel Bottom-up Dataset Generation Methodology**: We introduce a paradigm shift in optimization benchmarking dataset creation through `Schema2Opt`, which discovers optimization opportunities from database schemas rather than adapting known problems. This bottom-up approach analyzes existing information to identify latent decision variables, objective functions, and constraints, enabling the discovery of new optimization applications that may not exist in traditional operations research literature.
- **Comprehensive Benchmark Dataset**: We present `Schema2Opt`, a synthetic dataset specifically designed for database-grounded prescriptive analytics, along with a systematic generation framework that transforms SQL schemas into realistic optimization scenarios. Unlike existing business analytics benchmarks that provide high-level recommendations, `Schema2Opt` focuses on mathematical optimization problems with complete business contexts, operational databases, and verified solutions.
- **Effective Prescriptive Analytics Data Agent**: We develop `Data2Decision`, an effective prescriptive analytics data agent that employs a two-stage pipeline with test-time scaling and multi-solver consensus. This approach enables end-to-end automation without manual preprocessing and achieves up to 69.5% accuracy on `Schema2Opt`, outperforming existing approaches.

2 Related Work

2.1 From Text-to-SQL to Data Agents: Evolution and Limitations

Text-to-SQL systems have evolved significantly from rule-based approaches to neural architectures (Zhong et al., 2017; Yu et al., 2018), achieving impressive accuracy on benchmarks like Spider

2.0 (Lei et al., 2024), BIRD (Li et al., 2024a), and BIRD-CRITIC (Li et al., 2025). Data agents like InsightPilot (Ma et al., 2023) and DAgent (Xu et al., 2025) extend these capabilities by automating entire analytical workflows through multi-step reasoning and cross-table associations. However, both paradigms remain fundamentally limited to descriptive and diagnostic analytics, answering what happened rather than what actions to take. While InsightBench (Sahu et al., 2025) evaluates prescriptive tasks, its recommendations remain qualitative suggestions rather than quantitative solutions with verifiable optimal values. Neither approach can formulate or solve the constrained optimization problems essential for mathematically optimal decision-making in enterprise operations where complex trade-offs require mathematical rigor. Appendix A.1 provides extended discussion of these architectural limitations.

2.2 Text-to-OPT Methods and Prescriptive Analytics

The Text-to-OPT field has seen rapid development with ORLM (Tang et al., 2023) pioneering LLM use for operations research, followed by OptiMUS (Zheng et al., 2024) with iterative refinement, Chain-of-Experts (Xiao et al., 2023) with modular architectures, and SIRL (Chen et al., 2025) employing reinforcement learning with solver-based verification. However, all existing methods assume optimization parameters are embedded within problem descriptions as LaTeX tables or textbook scenarios, fundamentally misaligning with enterprise workflows where parameters must be extracted from operational databases. Furthermore, benchmarks like NL4OPT (Ramamonjison et al., 2022) and ComplexOR (Xiao et al., 2023) perpetuate a top-down paradigm, recycling textbook problems rather than discovering optimization opportunities that naturally emerge from real data structures. True prescriptive analytics requires bridging both data extraction and optimization solving (Lepenioti et al., 2020). Our work addresses this gap through `Schema2Opt`'s bottom-up approach for database-grounded prescriptive analytics, leveraging test-time scaling insights (Wang et al., 2022; Snell et al., 2024) with multi-solver consensus. Appendix A.2 examines these methods in detail, while Appendix A.3 discusses relevant test-time compute strategies.

3 Schema2Opt: Synthetic Data Generation Pipeline

Real-world prescriptive analytics fundamentally differs from traditional Text-to-OPT problems in how optimization parameters are obtained. While existing approaches assume these parameters are explicitly provided in problem descriptions, enterprise decision-making requires extracting them from operational databases through complex queries. We term this challenge **database-grounded prescriptive analytics** and formalize it as a function $f : (Q, S, D) \to A$, where Q represents natural language business objectives and constraints, S denotes the database schema with table structures, D contains the operational data, and $A = (x^*, v^*)$ comprises the optimal decisions and objective value. The function f embodies the complete prescriptive analytics pipeline: interpreting business requirements from Q, identifying relevant data sources through S, extracting parameters via queries against D, and solving the optimization model to produce A. Unlike traditional Text-to-OPT where parameter extraction is merely preprocessing, the database-grounded nature makes this transformation from (Q, S, D) to A an integral component that fundamentally shapes the optimization formulation itself.

The lack of realistic benchmarks for this database-grounded prescriptive analytics task stems from fundamental data sharing constraints. Organizations accumulate vast amounts of structured data in relational databases (Nambiar & Mundra, 2022), with data-driven practices growing rapidly across industries (Brynjolfsson & McElheran, 2016). However, the proprietary nature of enterprise data and embedded business logic makes sharing real-world optimization examples impossible due to governance and confidentiality requirements (Janssen et al., 2020). This creates a critical evaluation gap for systems that must bridge data extraction and optimization solving.

To address this challenge, we present `Schema2Opt`, a synthetic dataset that pairs enterprise database schemas with corresponding optimization problems. Unlike existing benchmarks that follow a top-down paradigm by starting with known optimization formulations, `Schema2Opt` introduces *the first bottom-up data generation approach that discovers optimization opportunities directly from database structures.* Our generation framework, shown in Figure 2, transforms Spider dataset schemas (Yu et al., 2018) through a five-stage pipeline. We utilize schemas from the Spider dataset with at most five tables to ensure tractability and generate two versions of the `Schema2Opt` dataset using different LLMs (`GPT-4o` and `DeepSeek-V3`) to ensure diversity and robustness.

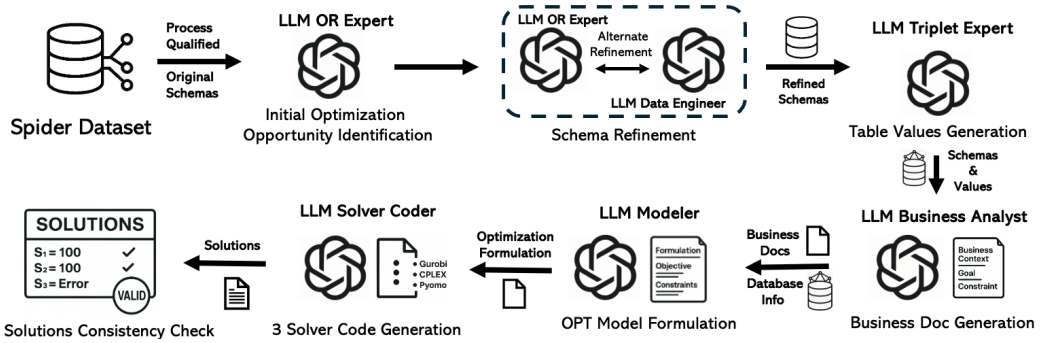

Figure 2: `Schema2Opt` synthetic data generation pipeline. Starting from Spider schemas, the framework iteratively refines schemas through OR Expert and Data Engineer dialogue, generates realistic data via Triple Expert collaboration, creates business documents, formulates optimization models, and validates solutions across multiple solvers.

Only instances that pass multi-solver majority-vote validation with consistent optimal values are included as valid cases in the final benchmark.

3.1 Schema Initialization and Analysis

We initialize `Schema2Opt` with schemas from the Spider dataset, a widely-used SQL benchmark containing database schemas designed to resemble real-world applications, filtered to databases with at most five tables. Unlike existing benchmarks that impose predetermined optimization problems, our bottom-up approach discovers optimization opportunities by analyzing how table structures naturally map to optimization parameters.

3.2 Alternating Expert Dialogue for Benchmark Generation

The core innovation of our framework lies in the alternating dialogue between specialized LLM-based agents that iteratively refine the optimization problem until convergence. Inspired by alternating optimization methods such as EM algorithms, our approach alternates between two complementary perspectives: *(1) the OR Expert agent refines optimization requirements while holding the schema fixed, then (2) the Data Engineer agent adjusts the schema to better support these requirements.* This iterative process ensures that optimization formulations and database structures co-evolve to achieve mutual consistency, similar to how alternating optimization methods achieve convergence by optimizing one component while fixing others. Appendix D provides the complete algorithmic specification and detailed prompt engineering strategies that enable this alternating optimization approach.

3.2.1 OR Expert Analysis

The OR Expert focuses exclusively on optimization modeling, evaluating how business objectives map to mathematical formulations while maintaining strict linearity constraints. Given the current schema, the expert identifies mapping adequacy for each optimization component and specifies missing requirements without proposing database changes.

> **OR Expert Guidance:** *Focus exclusively on optimization modeling and understanding current schema-to-optimization mapping. Design LINEAR optimization problems where objective function MUST be minimize/maximize $\sum$(coefficient $\times$ variable). Identify how optimization components map to current schema ...*

3.2.2 Data Engineer Implementation

The Data Engineer implements schema modifications based on the OR Expert's analysis, creating tables for decision variables, adjusting columns for coefficients, and managing business configuration

parameters. A key design principle is distinguishing between tabular data and scalar parameters: information that naturally forms multiple rows (e.g., product inventories, customer orders) belongs in database tables, while single-value parameters (e.g., daily capacity limits, minimum thresholds) and business formulas are stored in configuration files. This ensures proper data organization following database normalization principles.

> **Data Engineer Guidance:** *Implement schema changes following database normalization principles. Distinguish between tabular data and scalar configuration. Optimization information that represents collections belongs in database tables, while single-value parameters like capacity limits belong in configuration files. Create appropriate tables for decision variables and maintain business realism throughout the schema design ...*

This alternating process typically converges within 3-4 iterations (maximum 5), with an LLM-based judge evaluating mapping adequacy at each iteration to determine convergence, achieving complete mapping adequacy where all optimization components have identified data sources. Appendix D provides the complete algorithmic specification and detailed prompt engineering strategies that enable this alternating optimization approach.

3.3 Data Generation and Problem Formulation

After schema convergence through the alternating process described in Section 3.2, `Schema2Opt` generates realistic data and business descriptions. The framework employs an LLM acting as a triple expert with combined expertise in business operations, data management, and optimization modeling. This unified perspective ensures generated values reflect industry norms while maintaining cross-table consistency and enabling feasible solutions with meaningful trade-offs. An LLM-based business analyst then produces natural language descriptions that translate technical optimization requirements into business narratives, naturally incorporating configuration parameters without exposing their storage mechanism. To verify information completeness, we simulate how existing methods would solve each scenario. An LLM-based OR Expert attempts to extract optimization parameters from the business document and database to construct a complete mathematical model. Successfully producing a well-formed linear program validates that all necessary coefficients, constraints, and objectives can be derived from the provided data. Instances failing this verification are discarded, ensuring our benchmark contains only informationally complete problems. Appendix D provides detailed prompt engineering strategies for the triple expert and verification process.

3.4 Solution Generation and Verification

`Schema2Opt` ensures both mathematical correctness and practical solvability through a multi-stage verification process. *Template-guided generation* addresses the challenge of solver-specific API patterns that cause frequent errors in LLM-generated code. We provide templates for `Gurobipy`, `DOCplex`, and `Pyomo` that encapsulate best practices for variable declaration, constraint syntax, and result extraction, serving as in-context examples that significantly reduce syntax errors and API misuse. *Cross-solver validation* ensures solution reliability by executing all three solvers in parallel for each problem. We consider solutions valid only when at least two solvers agree on the optimal value within numerical tolerance or unanimously determine infeasibility. This majority voting guards against solver-specific numerical issues while maintaining high confidence in the ground truth values for our benchmark.

3.5 Dataset Format and Components

The resulting `Schema2Opt` *synthetic data generation framework represents a paradigm shift in optimization benchmark creation. Rather than recycling existing textbook problems or industrial case studies, our bottom-up approach enables automatic discovery of novel optimization opportunities that emerge naturally from enterprise data patterns.* Each case in `Schema2Opt` contains four core components: (1) a **business document** describing the decision scenario with context, goals and constraints, (2) a **database schema and data dictionary**, (3) **database content** and (4) **verified solutions**. We generate two dataset versions using `GPT-4o` (106 problems) and `DeepSeek-V3` (95 problems), validated through multi-solver consensus. Appendix B provides a comprehensive clas-

sification of the generated problems across business domains, optimization types, and complexity levels.

Illustrative Example: Inventory Optimization
Business Document: *Problem Context and Goals:* A retail company manages inventory across 3 warehouses to minimize total holding costs; *Constraints:* Daily capacity 1,000 units per warehouse, safety stock $\geq$ 20% demand; ...
Database Schema: `CREATE TABLE warehouses (id INT, capacity INT, cost FLOAT); CREATE TABLE products (id INT, holding_cost FLOAT, demand INT); ...`
Data Dictionary: `cost` $\rightarrow$ storage cost per unit ...
Database Content: `INSERT INTO warehouses VALUES (1, 1000, 2.5), (2, 1000, 3.0); INSERT INTO products VALUES (1, 5.0, 300), (2, 7.5, 450); ...`
Verified Solutions: Optimal value = $4,285.00

Appendix C presents six selected examples from both dataset versions that show the range of optimization scenarios discovered through our bottom-up generation method. The complete datasets generated by both `DeepSeek-V3` and `GPT-4o` are available in the supplementary materials for further exploration and analysis.

4 DATA2DECISION: A PRESCRIPTIVE ANALYTICS DATA AGENT FOR ENTERPRISE DECISION-MAKING

Having established the `Schema2Opt` data generation pipeline and benchmark datasets (Sec 3), we now present `Data2Decision`, the first data agent specifically designed for database-grounded prescriptive analytics. Unlike existing Text-to-OPT approaches that assume pre-embedded optimization parameters, `Data2Decision` must extract these parameters from enterprise databases before solving. Our system employs a two-stage pipeline: (1) analyzing business requirements to generate SQL queries that extract decision variables, objective coefficients, and constraint parameters from databases; (2) directly transforming the SQL-enhanced problem descriptions into executable optimization code. To address inherent uncertainties in both stages, we incorporate test-time scaling through self-consistency, temperature-controlled exploration, multi-solver diversification, and majority voting consensus. This design eliminates error-prone mathematical modeling steps while ensuring robust solutions through systematic exploration of diverse formulations. This design eliminates error-prone mathematical modeling steps while ensuring robust solutions through systematic exploration of diverse formulations. Appendix E details the complete implementation including SQL generation strategies and solver-specific code generation templates.

4.1 TEST-TIME SCALING THROUGH SELF-CONSISTENCY

Inspired by self-consistency approaches in reasoning tasks (Wang et al., 2022), we apply test-time scaling to prescriptive analytics by generating multiple diverse solutions and aggregating them through majority voting. This approach addresses the inherent uncertainties in database-grounded optimization: SQL queries may extract different subsets of relevant data, and the same optimization problem often admits multiple valid formulations. By sampling N independent solution attempts with controlled randomness and selecting the most frequent optimal value, we significantly improve robustness over single-attempt methods.

Given N parallel attempts producing optimal values, let $\mathcal{I}_{succ} \subseteq \{1, \ldots, N\}$ denote successful attempts with values $\{v_i : i \in \mathcal{I}_{succ}\}$. We employ majority voting:

$$v^* = \operatorname*{argmax}_v |\{i \in \mathcal{I}_{succ} : v_i = v\}|$$

where the argmax is taken over all unique values obtained. The consensus mechanism provides confidence estimates through agreement levels, with unanimous agreement indicating high confidence. We set $N = 10$ attempts per problem, which our parameter analysis shows effectively balances accuracy and efficiency.

4.2 TEMPERATURE-CONTROLLED EXPLORATION

We employ adaptive temperature scheduling across both pipeline stages to balance exploration and exploitation:

$$T_{sql}^{(i)} = T_{base}^{sql} + i \cdot \Delta T^{sql}, \quad T_{code}^{(i)} = T_{base}^{code} + i \cdot \Delta T^{code}$$

This progressive strategy serves different purposes at each stage. For SQL generation, temperature variation explores different interpretations of which data elements map to optimization components, addressing ambiguity when business requirements don't explicitly specify database relationships. For code generation, temperature diversity captures alternative valid formulations of the same optimization problem. In our experiments, we use wider temperature ranges for SQL generation ($T_{base}^{sql} = 0.05$, $\Delta T^{sql} = 0.05$) than code generation ($T_{base}^{code} = 0.0$, $\Delta T^{code} = 0.02$), reflecting their different uncertainty characteristics.

4.3 DIRECT OPTIMIZATION CODE GENERATION

A key design decision in `Data2Decision` is eliminating explicit mathematical modeling as an intermediate step. While conventional Text-to-OPT pipelines follow a three-stage process from natural language to mathematical formulation to executable code, we directly translate SQL-enhanced problem descriptions into solver-specific code. This approach is supported by recent findings in latent reasoning (Hao et al., 2024), which show that allowing models to reason implicitly in continuous latent space reduces hallucinations and errors compared to explicit step-by-step reasoning. We therefore bypass explicit mathematical formulation, allowing LLMs to leverage their internalized optimization knowledge directly. Our ablation study validates this design choice, showing significant performance degradation when introducing intermediate modeling steps.

4.4 MULTI-SOLVER DIVERSIFICATION

We further enhance solution robustness by cycling through `Gurobipy`, `DOCplex`, and `Pyomo` implementations across attempts. This strategy exploits the fact that LLMs have learned distinct modeling patterns from each solver's documentation and codebase during pre-training. By leveraging these varied modeling languages, we capture diverse optimization knowledge embedded in different solver languages, enabling broader problem coverage. This multi-solver approach complements temperature-based exploration: while temperature varies problem interpretations, solver diversity accesses different pre-trained modeling patterns, preventing framework-specific limitations from affecting solution quality.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Dataset. Two `Schema2Opt` test sets: `GPT-4o` (106 cases) and `DeepSeek-V3` (95 cases).

Evaluation Metrics. We focus on solution accuracy as the primary metric, measuring the percentage of problems where methods find the correct optimal solution.

Baselines and Implementation Details. We compare against two categories of approaches, both following a two-stage pipeline but with different setups. The `Schema2Opt` benchmark was generated using `GPT-4o` and `DeepSeek-V3`.

- **(i) Text-to-OPT methods with SQL assistance:** We evaluate `OR-LLM-Agent`, `Chain-of-Experts`, `OptiMUS 0.3`, and `ZeroShot` baseline. Since these methods cannot extract data from databases independently, we provide a **unified first stage**, using the corresponding backbone model (`GPT-4o-mini` for the `GPT-4o` test set, `DeepSeek-V3` for the `DeepSeek-V3` test set) to analyze business requirements, generate SQL queries, and extract optimization parameters from databases. This enhanced problem description containing both business context and retrieved data tables is then provided identically to all baselines for their second stage of optimization formulation.

Category	Method	DeepSeek-V3 Test Set (95 cases)		GPT-4o Test Set (106 cases)	
		Correct	Accuracy	Correct	Accuracy
Text-to-OPT with SQL	OR-LLM-Agent	62	65.3%	57	53.8%
	ZeroShot	53	55.8%	46	43.4%
	Chain-of-Experts	23	24.2%	21	19.8%
	OptiMUS 0.3 *	45	47.4%	3	2.8%
End-to-End Models	Llama-3.3-70B	51	53.7%	50	47.2%
	Qwen2.5-72B	24	25.3%	28	26.4%
	Phi-4	27	28.4%	22	20.8%
	Llama-4-Scout	5	5.3%	1	0.9%
Our Method	Data2Decision	**66**	**69.5%**	**57**	**53.8%**

Table 1: Performance comparison on Schema2Opt benchmark. *In OptiMUS 0.3, JSON errors cannot be reliably repaired by GPT-4o-mini, causing low performance.

- **(ii) End-to-end foundation models:** We evaluate Llama-3.3-70B, Qwen2.5-72B, Phi-4, and Llama-4-Scout, which handle **both stages independently**; they perform their own SQL extraction in the first stage and optimization formulation in the second stage using their respective models throughout.

Validation Details. We validate solutions by comparing each method's output against the ground truth optimal values from our dataset. To robustly extract results from diverse solver output formats, we use an LLM-based extraction with 5-attempt majority voting.

5.2 Main Results

Table 1 presents performance across both test sets. Our Data2Decision agent achieves the highest accuracy on both benchmarks, with 69.5% on DeepSeek-V3-generated problems and 53.8% on GPT-4o-generated problems. This represents a improvement over the best baselines in each category. Among Text-to-OPT methods with SQL assistance, OR-LLM-Agent performs best with 65.3% and 53.8% accuracy respectively, while Chain-of-Experts struggles at 24.2% and 19.8% despite having access to pre-extracted data. The performance of OptiMUS 0.3 varies dramatically between test sets, achieving 47.4% accuracy on DeepSeek-V3 problems but only 2.8% on GPT-4o problems, suggesting high sensitivity to problem structure. End-to-end foundation models face the additional challenge of joint SQL extraction and optimization formulation. Llama-3.3-70B demonstrates competitive performance at 53.7% and 47.2% accuracy, nearly matching specialized Text-to-OPT methods despite handling the complete pipeline. However, performance degrades significantly for other models, with Llama-4-Scout achieving only 5.3% and 0.9% accuracy. The consistent accuracy advantage of our approach across both test sets validates our two-stage pipeline design with test-time scaling, which effectively decomposes the complex task while maintaining robustness through multi-solver consensus.

The systematic performance gap between the two test sets across all methods reveals interesting dataset characteristics. The DeepSeek-V3-generated problems appear more amenable to optimization, with most methods achieving higher accuracy compared to their performance on GPT-4o-generated problems. This cross-dataset evaluation demonstrates the importance of diverse benchmark generation and highlights the generalization challenges in prescriptive analytics. Notably, our method maintains its relative advantage across both distributions, suggesting that the test-time scaling approach provides robustness beyond what single-attempt methods can achieve.

5.3 Ablation Study

We conduct comprehensive ablation studies on the DeepSeek-V3 test set to analyze the contribution of each component in our Data2Decision agent (Section 4). All experiments use 10 attempts as the baseline configuration unless otherwise specified.

5.3.1 Parameter Analysis

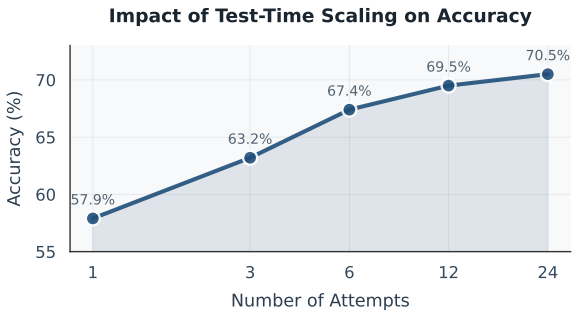

Figure 3: Impact of test-time scaling on accuracy. Performance demonstrates a clear scaling law, with accuracy improving monotonically as the number of attempts increases, exhibiting rapid initial gains that gradually plateau.

Figure 3 illustrates how the number of attempts affects performance in our test-time scaling approach. The accuracy curve shows steady improvement, rising from 57.9% with a single attempt to 63.2% with 3 attempts, 67.4% with 6 attempts, and reaching 69.5% at 12 attempts. Further increasing to 24 attempts yields minimal gains (70.5%), demonstrating clear diminishing returns. This scaling behavior confirms the effectiveness of our multi-attempt strategy while revealing a natural saturation point. *Our choice of 10 attempts (which would achieve approximately 69% accuracy based on the trend) effectively balances solution quality with computational efficiency, capturing most performance benefits while avoiding the computational overhead of larger attempt counts.*

5.3.2 Component Ablation

Table 2: Ablation study results on `DeepSeek-V3` test set.

Configuration	Correct	Accuracy
Multi-solver consensus	**66**	**69.5%**
Single solver (`Gurobipy` only)	50	52.6%
Incremental temperature	**66**	**69.5%**
Fixed low temperature	65	68.4%
Two-stage pipeline	**66**	**69.5%**
Three-stage pipeline	58	61.1%

Table 2 presents ablation results for key architectural components. (1) **Multi-solver consensus provides the largest performance gain.** Removing this component and relying on a single solver (`Gurobipy`) causes accuracy to drop from 69.5% to 52.6%, a substantial 16.9 percentage point decrease. This demonstrates that cross-validation across different solver implementations is crucial for avoiding suboptimal formulations caused by solver-specific biases. (2) **Temperature scheduling offers modest but consistent improvements.** Our incremental temperature strategy (0.05→0.50) outperforms fixed low-temperature generation (0.01) by 1.1 percentage points (69.5% vs 68.4%). While the gain is smaller than other components, the progressive exploration from conservative to diverse generation helps discover alternative valid formulations. (3) **The two-stage pipeline architecture is superior to more complex alternatives.** Introducing an intermediate mathematical modeling stage in a three-stage pipeline reduces accuracy from 69.5% to 61.1%. This 8.4 percentage point drop reveals that additional abstraction layers increase error propagation without providing compensating benefits, validating our streamlined design that directly translates SQL-extracted data to executable optimization code.

6 Conclusion

In this paper, we introduced database-grounded prescriptive analytics tasks, where optimization parameters must be extracted from enterprise databases rather than being explicitly provided. To address this task, we presented `Schema2Opt` (Sec 3), a comprehensive benchmark for this problem, and proposed `Data2Decision` (Sec 4), a two-stage data agent system with test-time scaling that achieves strong performance on both test sets. Our bottom-up data generation approach represents a paradigm shift from recycling textbook problems to discovering optimization opportunities inherent in database structures. This work bridges the gap between descriptive analytics and mathematical optimization, enabling end-to-end prescriptive analytics for enterprise decision-making. Future work could explore developing models that can better handle the joint challenges of data extraction and optimization formulation.

Ethics Statement

This work introduces a synthetic benchmark generation framework that transforms existing database schemas into optimization problems. All source data originates from the publicly available Spider dataset (Yu et al., 2018), which contains no personal or sensitive information. Our framework generates entirely synthetic business scenarios and numerical data without utilizing any real enterprise information, thus avoiding potential privacy concerns that arise in real-world business analytics (Janssen et al., 2020).

Reproducibility Statement

To ensure full reproducibility, we provide relevant code and both datasets generated by `DeepSeek-V3` and `GPT-4o` in the supplementary materials. Each dataset is organized within the `schema2opt_[model]` directory structure, with individual database problems stored under `/schema2optsgd/text2opt_dataset_alternating_opt/database_name/`. For complete problem-to-solution cases, refer to `problem_solution_description.md` files that contain the full pipeline from problem specification to verified solution. The specific prompts for different agent roles can be found in the `/schema2optsgd/templates/` folder within each dataset directory. This includes the complete `Schema2Opt` generation pipeline with prompts and agent specifications detailed in Appendix D, the `Data2Decision` implementation with SQL generation strategies and solver templates described in Appendix E, and both versions (`GPT-4o` and `DeepSeek-V3`) of the `Schema2Opt` benchmark. For optimization solving, we employ three modeling frameworks: `Gurobipy` (Python API for Gurobi solver), `DOCplex` (IBM's Python modeling API), and `Pyomo` (open-source Python modeling framework), with Gurobi 12.0.2 and IBM CPLEX 22.1.2 as the underlying optimization solvers.

References

Ali AhmadiTeshnizi, Wenzhi Gao, Herman Brunborg, Shayan Talaei, Connor Lawless, and Madeleine Udell. Optimus-0.3: Using large language models to model and solve optimization problems at scale. *arXiv preprint arXiv:2407.19633*, 2024.

Nicolás Astorga, Tennison Liu, Yuanzhang Xiao, and Mihaela van der Schaar. Autoformulation of mathematical optimization models using llms. *arXiv preprint arXiv:2411.01679*, 2024.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 17682–17690, 2024.

Zhenni Bi, Kai Han, Chuanjian Liu, Yehui Tang, and Yunhe Wang. Forest-of-thought: Scaling test-time compute for enhancing llm reasoning. *arXiv preprint arXiv:2412.09078*, 2024.

Erik Brynjolfsson and Kristina McElheran. The rapid adoption of data-driven decision-making. *American Economic Review*, 106(5):133–139, 2016.

Xuanhe Chen, Ji Zhang, Guoliang Xiong, and Jianhua Li. Towards automated data integration and optimization for database systems. *Proceedings of the VLDB Endowment*, 13(12):3366–3369, 2020.

Yitian Chen, Jingfan Xia, Siyu Shao, Dongdong Ge, and Yinyu Ye. Solver-informed rl: Grounding large language models for authentic optimization modeling. *arXiv preprint arXiv:2505.11792*, 2025.

Thomas H. Davenport and Jeanne G. Harris. *Competing on Analytics: Updated, with a New Introduction: The New Science of Winning*. Harvard Business Review Press, 2017.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

Zijin Hong, Zheng Yuan, Qinggang Zhang, Hao Chen, Junnan Dong, Feiran Huang, and Xiao Huang. Next-generation database interfaces: A survey of llm-based text-to-sql. *arXiv preprint arXiv:2406.08426*, 2024.

Chenyu Huang, Zhengyang Tang, Shixi Hu, Ruoqing Jiang, Xin Zheng, Dongdong Ge, Benyou Wang, and Zizhuo Wang. Orlm: A customizable framework in training large models for automated optimization modeling. *Operations Research*, 2025.

Siful Islam. Future trends in sql databases and big data analytics: Impact of machine learning and artificial intelligence. *Available at SSRN 5064781*, 2024.

Marijn Janssen, Paul Brous, Elsa Estevez, Luis S Barbosa, and Tomasz Janowski. Data governance: Organizing data for trustworthy artificial intelligence. *Government information quarterly*, 37(3): 101493, 2020.

Caigao Jiang, Xiang Shu, Hong Qian, Xingyu Lu, Jun Zhou, Aimin Zhou, and Yang Yu. Llmopt: Learning to define and solve general optimization problems from scratch. *arXiv preprint arXiv:2410.13213*, 2024.

George Katsogiannis-Meimarakis and Georgia Koutrika. A survey on deep learning approaches for text-to-sql. *The VLDB Journal*, 32(4):905–936, 2023.

Sascha Kraus, Paul Jones, Norbert Kailer, Alexandra Weinmann, Nuria Chaparro-Banegas, and Norat Roig-Tierno. Digital transformation: An overview of the current state of the art of research. *Sage Open*, 11(3):21582440211047576, 2021.

Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, et al. Spider 2.0: Evaluating language models on real-world enterprise text-to-sql workflows. *arXiv preprint arXiv:2411.07763*, 2024.

Katerina Lepenioti, Alexandros Bousdekis, Dimitris Apostolou, and Gregoris Mentzas. Prescriptive analytics: Literature review and research challenges. *International Journal of Information Management*, 50:57–70, 2020.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36, 2024a.

Jinyang Li, Nan Huo, Yan Gao, Jiayi Shi, Yingxiu Zhao, Ge Qu, Yurong Wu, Chenhao Ma, Jian-Guang Lou, and Reynold Cheng. Tapilot-crossing: Benchmarking and evolving llms towards interactive data analysis agents. *arXiv preprint arXiv:2403.05307*, 2024b.

Jinyang Li, Xiaolong Li, Ge Qu, Per Jacobsson, Bowen Qin, Binyuan Hui, Shuzheng Si, Nan Huo, Xiaohan Xu, Yue Zhang, et al. Swe-sql: Illuminating llm pathways to solve user sql issues in real-world applications. *arXiv preprint arXiv:2506.18951*, 2025.

Vinicius Lima, Dzung T Phan, Jayant Kalagnanam, Dhaval Patel, and Nianjun Zhou. Toward a trustworthy optimization modeling agent via verifiable synthetic data generation. *arXiv preprint arXiv:2508.03117*, 2025.

Hongliang Lu, Zhonglin Xie, Yaoyu Wu, Can Ren, Yuxuan Chen, and Zaiwen Wen. OptMATH: A scalable bidirectional data synthesis framework for optimization modeling. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=9P5e6iE4WK.

Pingchuan Ma, Rui Ding, Shuai Wang, Shi Han, and Dongmei Zhang. Demonstration of insightpilot: An llm-empowered automated data exploration system. *arXiv preprint arXiv:2304.00477*, 2023.

Andrew McAfee, Erik Brynjolfsson, Thomas H Davenport, DJ Patil, and Dominic Barton. Big data: the management revolution. *Harvard business review*, 90(10):60–68, 2012.

Athira Nambiar and Divyansh Mundra. An overview of data warehouse and data lake in modern enterprise data management. *Big data and cognitive computing*, 6(4):132, 2022.

Rindranirina Ramamonjison, Timothy Yu, Raymond Li, Haley Li, Giuseppe Carenini, Bissan Ghaddar, Shiqi He, Mahdi Mostajabdaveh, Amin Banitalebi-Dehkordi, Zirui Zhou, and Yong Zhang. Nl4opt competition: Formulating optimization problems based on their natural language descriptions. In Marco Ciccone, Gustavo Stolovitzky, and Jacob Albrecht (eds.), *Proceedings of the NeurIPS 2022 Competitions Track*, volume 220 of *Proceedings of Machine Learning Research*, pp. 189–203. PMLR, 28 Nov–09 Dec 2022. URL https://proceedings.mlr.press/v220/ramamonjison23a.html.

Gaurav Sahu, Abhay Puri, Juan Rodriguez, Amirhossein Abaskohi, Mohammad Chegini, Alexandre Drouin, Perouz Taslakian, Valentina Zantedeschi, Alexandre Lacoste, David Vazquez, et al. InsightBench: Evaluating business analytics agents through multi-step insight generation. In *International Conference on Learning Representations (ICLR)*, 2025.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Wei Sun, Scott McFaddin, Linh Ha Tran, Shivaram Subramanian, Kristjan Greenewald, Yeshi Tenzin, Zack Xue, Youssef Drissi, and Markus Ettl. PresAIse, a prescriptive ai solution for enterprise. *INFOR: Information Systems and Operational Research*, 62(4):629–645, 2024.

Xiaodong Tang, Jie Wang, and Xiaolei Chen. Large language models for operations research: The next frontier in optimization modeling, 2023.

Caigao Wang, Qiang Li, Runzhong Zhang, and Yu Qiao. LLM-OPT: Learning to define and solve general optimization problems from scratch, 2024.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Luoxuan Weng, Xingbo Wang, Junyu Lu, Yingchaojie Feng, Yihan Liu, and Wei Chen. Insightlens: Discovering and exploring insights from conversational contexts in large-language-model-powered data analysis. *CoRR*, 2024.

Christopher Wissuchek and Patrick Zschech. Prescriptive analytics systems revised: a systematic literature review from an information systems perspective. *Information Systems and e-Business Management*, pp. 1–75, 2024.

Ziyang Xiao, Dongxiang Zhang, Yangjun Wu, Lilin Xu, Yuan Jessica Wang, Xiongwei Han, Xiaojin Fu, Tao Zhong, Jia Zeng, Mingli Song, et al. Chain-of-experts: When llms meet complex operations research problems. In *The twelfth international conference on learning representations*, 2023.

Wenyi Xu, Yuren Mao, Xiaolu Zhang, Chao Zhang, Xuemei Dong, Mengfei Zhang, and Yunjun Gao. Dagent: A relational database-driven data analysis report generation agent. *arXiv preprint arXiv:2503.13269*, 2025.

Zhicheng Yang, Yiwei Wang, Yinya Huang, Zhijiang Guo, Wei Shi, Xiongwei Han, Liang Feng, Linqi Song, Xiaodan Liang, and Jing Tang. Optibench meets resocratic: Measure and improve LLMs for optimization modeling. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=fsDZwS49uY.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3911–3921, 2018.

Bowen Zhang and Pengcheng Luo. Or-llm-agent: Automating modeling and solving of operations research optimization problem with reasoning large language model. *arXiv preprint arXiv:2503.10009*, 2025.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301*, 2025.

Ali Zheng, Omar Arshad, Anton J. Kleywegt, and Alejandro Toriello Zvara. OptiMUS: Optimization modeling using mip solvers and large language models, 2024.

Victor Zhong, Caiming Xiong, and Richard Socher. Seq2SQL: Generating structured queries from natural language using reinforcement learning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1–11, 2017.

Appendices

A Extended Related Work

A.1 From Text-to-SQL to Data Agents: Evolution and Limitations

Text-to-SQL systems have evolved significantly from rule-based approaches to neural architectures (Zhong et al., 2017; Yu et al., 2018), with recent advances leveraging LLMs for improved semantic parsing (Hong et al., 2024; Katsogiannis-Meimarakis & Koutrika, 2023; Chen et al., 2020). These systems have achieved impressive accuracy on benchmarks like Spider 2.0 (Lei et al., 2024), BIRD (Li et al., 2024a), and BIRD-CRITIC (Li et al., 2025), enabling natural language interfaces for database querying. Building upon these foundations, data agents represent the next evolutionary step in automated data analysis. Systems like InsightPilot (Ma et al., 2023) pioneered LLM-empowered data exploration with goal-oriented querying capabilities. DAgent (Xu et al., 2025) advances this further by generating comprehensive analytical reports through multi-step reasoning and cross-table associations. Interactive approaches such as Tapilot-Crossing (Li et al., 2024b) incorporate self-reflection strategies to evolve agent capabilities, while InsightLens (Weng et al., 2024) integrates code, visualization, and natural language for multi-modal insight management. These agents substantially extend Text-to-SQL capabilities by automating the entire analytical workflow rather than just query generation.

However, both Text-to-SQL systems and data agents remain fundamentally limited to descriptive and diagnostic analytics. They excel at answering what happened and why it happened, but cannot

determine optimal actions through mathematical optimization. While recent benchmarks like InsightBench (Sahu et al., 2025) evaluate agents on prescriptive tasks, the resulting recommendations remain qualitative suggestions such as "open an incident ticket" rather than quantitative solutions with verifiable optimal values. Neither approach can formulate or solve the constrained optimization problems essential for mathematically optimal decision-making in enterprise operations.

A.2 Text-to-OPT Methods and Prescriptive Analytics

The Text-to-OPT field has seen rapid development with various approaches addressing mathematical optimization from natural language. ORLM (Tang et al., 2023) pioneered the use of LLMs for optimization modeling in operations research, while subsequent methods like Chain-of-Experts (Xiao et al., 2023) introduced multi-component frameworks with domain-specific modules orchestrated by a conductor. OptiMUS (Zheng et al., 2024) further advanced the field with a modular system capable of iterative model refinement and debugging. LLMOPT (Jiang et al., 2024) proposes a unified learning-based framework with a five-element formulation. OR-LLM-Agent (Zhang & Luo, 2025) leverages reasoning LLMs through task decomposition into several stages. Autoformulation (Astorga et al., 2024) introduces Monte Carlo Tree Search to systematically explore the formulation space, while SIRL (Chen et al., 2025) employs reinforcement learning with solver-based verification. Recent work on OptiTrust (Lima et al., 2025) introduces verifiable synthetic data generation pipelines for training trustworthy optimization modeling agents.

Despite these advances, all existing Text-to-OPT methods share a critical limitation: they assume that all necessary data is embedded within problem descriptions, typically in the form of LaTeX tables or natural language specifications. This assumption fundamentally misaligns with real-world enterprise scenarios where optimization parameters must be extracted from large databases. Furthermore, existing benchmarks like NL4OPT (Ramamonjison et al., 2022), ComplexOR (Xiao et al., 2023), and IndustryOR (Huang et al., 2025) perpetuate a problematic top-down data generation paradigm. These datasets are created by starting with well-known optimization problems from textbooks or literature, then generating variations through parameter adjustments or constraint modifications. This approach confines the field to recycling existing formulations rather than discovering optimization opportunities that naturally emerge from enterprise data structures.

True prescriptive analytics requires bridging this gap between data extraction and optimization solving (Lepenioti et al., 2020). Unlike Text-to-SQL which retrieves data from databases, or Text-to-OPT which assumes pre-extracted parameters, prescriptive analytics must accomplish both: extracting optimization parameters from databases and solving for optimal decisions. While recent solutions like PresAIse (Sun et al., 2024) combine causal inference with optimization, and InsightBench (Sahu et al., 2025) evaluates prescriptive tasks, these approaches provide only high-level suggestions. This gap motivates our development of `Schema2Opt` for rigorous evaluation of database-grounded prescriptive analytics.

A.3 Test-time Scaling for Complex Reasoning

Recent work has shown that Large Language Model (LLM) performance can be improved through test-time scaling, such as generating intermediate reasoning steps (Wei et al., 2022; Yao et al., 2023; Besta et al., 2024). Self-consistency (Wang et al., 2022) also shows that sampling multiple reasoning paths and selecting the most frequent answer enhances reliability on complex reasoning tasks. More recent advances demonstrate that test-time compute scaling can be more effective than simply increasing model parameters (Snell et al., 2024). Methods like Forest-of-Thought (Bi et al., 2024) extend tree-based reasoning to explore multiple solution paths simultaneously, enabling backtracking and lookahead capabilities. Process-based reward models have shown particular promise in verifying intermediate reasoning steps (Zhang et al., 2025), improving both the quality and reliability of generated solutions. These approaches have proven particularly effective for mathematical reasoning and code generation, with compute-optimal strategies achieving over 4x efficiency improvements compared to best-of-N baselines.

B Dataset Classification and Analysis

To understand the characteristics and diversity of optimization problems in Schema2Opt, we analyzed all generated instances before multi-solver validation. This comprehensive classification employed GPT-5.1-mini to evaluate each problem across five key dimensions: business domain, optimization type, problem complexity, implementation difficulty, and real-world applicability score.

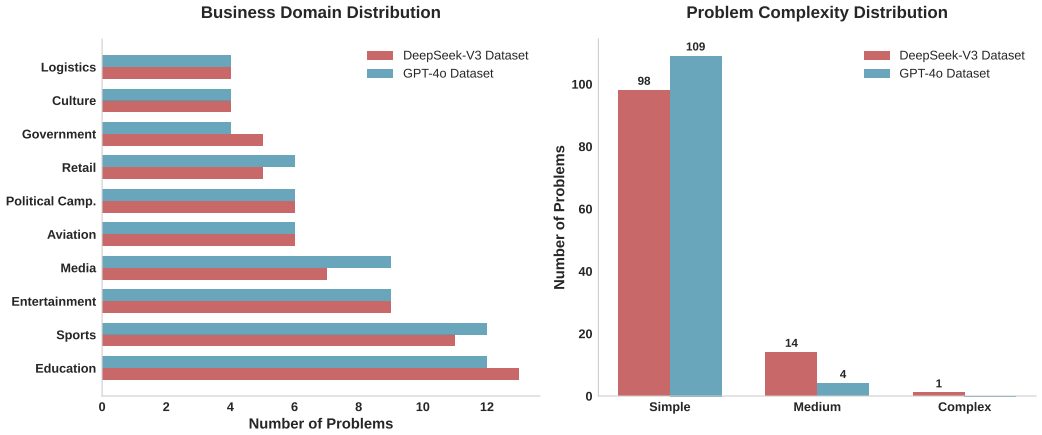

Figure 4: Business domain distribution and problem complexity analysis across DeepSeek-V3 and GPT-4o generated datasets.

The classification results reveal distinct patterns in problem generation. Our analysis identified 37 unique business sectors across both datasets, including education (25), sports (23), entertainment (18), media (16), aviation (12), political campaigns (12), retail (11), government (9), culture (8), logistics (8), manufacturing (8), hospitality (7), retail banking (6), college athletics (6), finance (5), technology (5), events and wedding planning (4), maritime shipping (4), academic conferences (3), academic institutions (3), disaster response (3), agriculture (2), construction (2), energy (2), gaming (2), healthcare (2), research (2), winemaking (2), and several specialized domains appearing once such as customer service centers, performing arts production, streaming systems, and transportation operations. Figure 4 visualizes the top 10 domains, showing how education, sports, and entertainment lead the distribution for both DeepSeek-V3 and GPT-4o datasets.

The complexity analysis reveals a strong preference for tractable problems. With 86.7% of DeepSeek-V3 problems and 96.5% of GPT-4o problems classified as simple, our bottom-up generation approach appears to naturally discover linear programming formulations that are computationally feasible. Only 14 DeepSeek-V3 problems and 4 GPT-4o problems reach medium complexity, while a single complex problem appears in the DeepSeek-V3 dataset. This distribution suggests that database schemas inherently encode optimization opportunities that align well with standard solver capabilities.

Figure 5 illustrates another consistent pattern: both datasets favor maximization over minimization objectives. The DeepSeek-V3 dataset contains 67.3% maximization problems while GPT-4o reaches 75.2%, reflecting how businesses naturally frame goals around maximizing revenue, efficiency, or satisfaction. After rigorous multi-solver validation, 95 problems from the DeepSeek-V3 generation and 106 from GPT-4o met all verification criteria to form the final Schema2Opt benchmark. This filtering ensures that every included problem represents not only a realistic business scenario but also produces consistent, verifiable optimal solutions across multiple solver implementations.

C Representative Dataset Examples from Schema2Opt

We present six complete examples from Schema2Opt that illustrate the breadth of optimization scenarios our bottom-up generation approach discovers. These examples, selected from both the GPT-4o and DeepSeek-V3 generated datasets, showcase how database schemas naturally en-

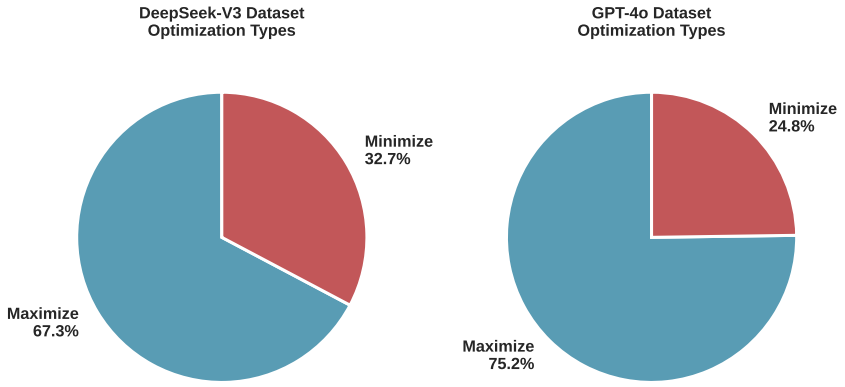

Figure 5: Distribution of optimization objectives showing preference for maximization problems in both datasets.

code diverse business decision problems spanning education, sports, entertainment, and logistics domains. Each example demonstrates the complete transformation from an existing SQL schema through our alternating expert dialogue to a verified optimization solution.

The examples reveal several key characteristics of our generation framework. First, they show how the same underlying database structure can support different optimization objectives through our iterative refinement process. For instance, Examples 1 and 2 both involve university activity data but diverge into distinct optimization problems focused on participation versus scoring. Second, they demonstrate the framework's ability to handle varying complexity levels, from simple linear programs with a handful of variables to more complex scheduling problems with multiple constraint types. Finally, these examples validate our design principle of separating tabular data from scalar configuration parameters, showing how business rules naturally emerge as configuration values while entity-specific data resides in database tables.

C.1 Example 1: University Activity Allocation (DeepSeek-V3-Generated)

This example demonstrates how student and faculty participation constraints naturally map to binary allocation decisions with capacity limitations.

Problem Context, Goals and Constraints

Context
A university is managing the allocation of students and faculty to extracurricular activities with the goal of maximizing overall participation. The decision-making process involves determining which students and faculty members should participate in which activities. Each student can participate in at most one activity, and each faculty member can participate in at most two activities. Additionally, each activity has a predefined maximum number of participants that cannot be exceeded.
The business configuration includes the following operational parameters: **Faculty Availability Limit**: Faculty members are limited to participating in a maximum of two activities to balance their workload and ensure availability. **Student Preference Threshold**: Students are allowed to participate in only one activity to ensure focused engagement and avoid overcommitment. **Total Participation Calculation**: The total participation in an activity is calculated as the sum of student and faculty participation in that activity.
The optimization problem is designed to ensure that these constraints are respected while maximizing the total number of participants across all activities.
Goals
The primary goal of this optimization problem is to maximize the total participation in extracurricular activities by both students and faculty. Success is measured by the total number of participants across all activities, which is the sum of student and faculty participation. This goal aligns with the operational parameters and ensures that the allocation respects the constraints on faculty availability, student preferences, and activity capacity limits.
Constraints
The optimization problem must adhere to the following constraints: 1. **Student Participation Limit**: Each student can participate in at most one activity. This ensures that students are not overcommitted and can focus on their chosen activity. 2. **Faculty Participation Limit**: Each faculty member can participate in at most two activities. This constraint balances faculty workload and ensures their availability across activities. 3. **Activity Capacity Limit**: The total number of participants in each activity, including both students and faculty, must not exceed the predefined maximum capacity for that activity. This ensures that activities are not overcrowded and can operate effectively.

These constraints are designed to ensure that the allocation of participants is feasible and aligns with the operational capabilities of the university.

Available Data - Database Schema

```
CREATE TABLE Participates_in (
  stuid INTEGER,
  actid INTEGER
);

CREATE TABLE Faculty_Participates_in (
  FacID INTEGER,
  actid INTEGER
);

CREATE TABLE Activity_Capacity (
  actid INTEGER,
  max_participants INTEGER
);

INSERT INTO Participates_in (stuid, actid) VALUES (101, 1);
INSERT INTO Participates_in (stuid, actid) VALUES (102, 2);
INSERT INTO Participates_in (stuid, actid) VALUES (103, 3);

INSERT INTO Faculty_Participates_in (FacID, actid) VALUES (201, 1);
INSERT INTO Faculty_Participates_in (FacID, actid) VALUES (202, 2);
INSERT INTO Faculty_Participates_in (FacID, actid) VALUES (203, 3);

INSERT INTO Activity_Capacity (actid, max_participants) VALUES (1, 10);
INSERT INTO Activity_Capacity (actid, max_participants) VALUES (2, 15);
INSERT INTO Activity_Capacity (actid, max_participants) VALUES (3, 20);
```

Data Dictionary - **Participates_in**: Tracks student participation in activities. - **stuid**: Unique identifier for a student. Used to determine which students participate in which activities. - **actid**: Unique identifier for an activity. Used to link students to specific activities. - **Faculty_Participates_in**: Tracks faculty participation in activities. - **FacID**: Unique identifier for a faculty member. Used to determine which faculty members participate in which activities. - **actid**: Unique identifier for an activity. Used to link faculty members to specific activities. - **Activity_Capacity**: Defines the maximum number of participants allowed in each activity. - **actid**: Unique identifier for an activity. Used to link capacity limits to specific activities. - **max_participants**: The maximum number of participants allowed in the activity. Used to enforce capacity constraints.

Mathematical Optimization Formulation

Decision Variables

- $x_{s,a}$: Binary variable indicating whether student s participates in activity a (1 if yes, 0 otherwise).
- $y_{f,a}$: Binary variable indicating whether faculty f participates in activity a (1 if yes, 0 otherwise).

Objective Function

Maximize the total participation across all activities:

$$\text{Maximize} \sum_{s} \sum_{a} x_{s,a} + \sum_{f} \sum_{a} y_{f,a}$$

Coefficients: All coefficients are 1, as each participant (student or faculty) contributes equally to the total participation.

Constraints

1. Student Participation Limit: $\sum_{a} x_{s,a} \leq 1 \quad \forall s$ (1)
2. Faculty Participation Limit: $\sum_{a} y_{f,a} \leq 2 \quad \forall f$ (2)
3. Activity Capacity Limit: $\sum_{s} x_{s,a} + \sum_{f} y_{f,a} \leq \text{max\_participants}_{a} \quad \forall a$ (3)
4. Binary Constraints: $x_{s,a} \in \{0,1\} \quad \forall s,a; \quad y_{f,a} \in \{0,1\} \quad \forall f,a$ (4)

Cross-Solver Analysis and Final Recommendation

Solver Results Comparison

Solver	Status	Optimal Value	Execution Time
Gurobipy	OPTIMAL	9.00	0.18s
Docplex	OPTIMAL	9.00	1.13s
Pyomo	OPTIMAL	9.00	0.94s

Solver Consistency Analysis

Result: All solvers produced consistent results ✓

Consistent Solvers: gurobipy, docplex, pyomo
Majority Vote Optimal Value: 9.0
Final Recommendation
Recommended Optimal Value: 9.0
Confidence Level: HIGH
Preferred Solver(s): gurobipy
Reasoning: Gurobipy is recommended due to its fastest execution time while still providing the optimal solution.

C.2 Example 2: Faculty Activity Optimization (GPT-4o-Generated)

Here the framework (GPT-4o as the backend LLM) transforms the same domain into a scoring optimization problem, showing how different business objectives emerge from similar database structures through our iterative refinement process.

Problem Context, Goals and Constraints

Context
The university is focused on optimizing the allocation of faculty members to various activities to enhance the overall participation scores. The decision-making process involves determining whether a faculty member, identified by their unique ID, should be assigned to a specific activity. This decision is represented by binary variables, where each variable indicates if a faculty member is assigned to an activity. The primary objective is to maximize the total participation score, which is calculated by summing the product of participation scores for each faculty-activity pair and the corresponding binary decision variable.
Operational parameters are crucial in this context. Each faculty member has a maximum number of activities they can participate in, ensuring they are not overburdened. Additionally, each activity requires a minimum number of faculty members to ensure it is adequately staffed. These parameters are derived from the business configuration, which includes the maximum number of activities a faculty member can participate in and the minimum number of faculty members required for an activity. The problem is structured to ensure that these constraints are respected, leading to a linear optimization formulation.
Goals
The primary goal of this optimization problem is to maximize the total participation score. This involves assigning faculty members to activities in a way that the sum of the participation scores for all faculty-activity assignments is maximized. The success of this optimization is measured by the total participation score achieved, which directly correlates with the participation scores assigned to each faculty-activity pair. The goal is articulated in natural language to emphasize the linear nature of the optimization objective.
Constraints
The optimization problem is subject to several constraints that ensure the feasibility and practicality of the solution: - Each faculty member can participate in a limited number of activities, as defined by their availability. This constraint ensures that the sum of the binary decision variables for each faculty member does not exceed their maximum availability. - Each activity must have a minimum number of faculty members assigned to it. This constraint ensures that the sum of the binary decision variables for each activity meets or exceeds the required staffing level.

Available Data - Database Schema

```
CREATE TABLE Participation_Score (
  FacID INTEGER,
  actid INTEGER,
  participation_score FLOAT
);

CREATE TABLE Faculty_Participates_in (
  FacID INTEGER,
  actid INTEGER,
  participation_score FLOAT
);

INSERT INTO Participation_Score (FacID, actid, participation_score) VALUES (1, 101, 12.0);
INSERT INTO Participation_Score (FacID, actid, participation_score) VALUES (2, 102, 18.5);
INSERT INTO Participation_Score (FacID, actid, participation_score) VALUES (3, 103, 14.0);

INSERT INTO Faculty_Participates_in (FacID, actid, participation_score) VALUES (1, 101, 12.0);
INSERT INTO Faculty_Participates_in (FacID, actid, participation_score) VALUES (2, 102, 18.5);
INSERT INTO Faculty_Participates_in (FacID, actid, participation_score) VALUES (3, 103, 14.0);
INSERT INTO Faculty_Participates_in (FacID, actid, participation_score) VALUES (1, 102, 10.0);
INSERT INTO Faculty_Participates_in (FacID, actid, participation_score) VALUES (2, 103, 16.0);
```

Data Dictionary
The data dictionary provides a comprehensive mapping of tables and columns to their business purposes and optimization roles:
- **Participation_Score Table**: This table stores the participation scores for each faculty-activity pair. The participation score represents the benefit of assigning a specific faculty member to an activity. The table includes: - **FacID**: Represents the unique identifier for each faculty member. - **actid**: Represents the unique identifier for each activity. - **participation_score**: Represents the score associated with assigning a faculty member to an activity, serving as a coefficient in the objective function.
- **Faculty_Participates_in Table**: This table tracks the participation of faculty members in activities. It includes: - **FacID**: Represents the unique identifier for each faculty member. - **actid**: Represents the unique identifier for each activity. - **participation_score**: Although included, this column is primarily used for tracking purposes and aligns with the participation scores in the Participation_Score table.

Mathematical Optimization Formulation

Decision Variables - Let $x_{i,j}$ be a binary decision variable where $x_{i,j} = 1$ if faculty member i is assigned to activity j, and $x_{i,j} = 0$ otherwise.
Objective Function
Maximize the total participation score:

$$\text{Maximize} \sum_{i,j} \text{Participation\_Score}(i,j) \times x_{i,j}$$

Where: - $\text{Participation\_Score}(i,j)$ is the participation score for faculty member i and activity j from the `Participation_Score` table.
Constraints

1. Faculty Availability Constraint: Each faculty member can participate in a limited number of activities. (5)

$$\sum_{j} x_{i,j} \leq \text{Max\_Activities}(i) \quad \forall i \tag{6}$$

- $\text{Max\_Activities}(i)$ is the maximum number of activities faculty member i can participate in, (7)

derived from business configuration. (8)

2. Activity Staffing Constraint: Each activity must have a minimum number of faculty members assigned. (9)

$$\sum_{i} x_{i,j} \geq \text{Min\_Faculty}(j) \quad \forall j \tag{10}$$

- $\text{Min\_Faculty}(j)$ is the minimum number of faculty members required for activity j, (11)

derived from business configuration. (12)

3. Binary Constraint: Each decision variable is binary. (13)

$$x_{i,j} \in \{0,1\} \quad \forall i,j \tag{14}$$

Data Source Verification: - Participation scores $\text{Participation\_Score}(i,j)$ are sourced from the `Participation_Score` table. - Maximum activities per faculty $\text{Max\_Activities}(i)$ and minimum faculty per activity $\text{Min\_Faculty}(j)$ are derived from business configuration parameters, which are not explicitly detailed in the provided data but are assumed to be part of the business rules. This formulation provides a complete, immediately solvable linear mathematical model using the given data and constraints.

Cross-Solver Analysis and Final Recommendation

Solver Results Comparison

Solver	Status	Optimal Value	Execution Time
Gurobipy	ERROR	N/A	0.18s
Docplex	OPTIMAL	70.50	1.39s
Pyomo	OPTIMAL	70.50	1.12s

Solver Consistency Analysis
Result: All solvers produced consistent results ✓
Consistent Solvers: docplex, pyomo
Majority Vote Optimal Value: 70.5
Final Recommendation
Recommended Optimal Value: 70.5
Confidence Level: HIGH
Preferred Solver(s): docplex/pyomo
Reasoning: Both DOCplex and Pyomo provided consistent and optimal results, indicating reliability. Gurobipy's error suggests data issues that need addressing before it can be considered.

The next two examples demonstrate how our framework handles optimization in the sports domain, specifically bodybuilder team selection and performance optimization. These cases show how physical attributes and performance metrics stored in separate database tables can be unified into cohesive optimization models through our schema refinement process.

C.3 Example 3: Bodybuilder Team Selection (DeepSeek-V3-Generated)

This example shows the integration of performance metrics with physical constraints, where team composition must balance total scoring against average height and weight requirements.

Problem Context, Goals and Constraints

Context
A bodybuilding competition organizer is tasked with selecting a team of bodybuilders to compete in an upcoming event. The goal

is to assemble a team that maximizes the total performance score based on the bodybuilders' Snatch and Clean & Jerk scores. The selection process must adhere to specific operational constraints to ensure the team meets diversity and physical criteria.
The organizer must decide which bodybuilders to include in the team, represented by a binary decision for each individual. The total number of bodybuilders in the team cannot exceed a predefined limit, ensuring the team remains manageable and diverse. Additionally, the team must meet a minimum average height requirement of 170 cm and a maximum average weight requirement of 100 kg. These constraints ensure the team aligns with the competition's physical standards.
The performance scores for each bodybuilder are derived from their Snatch and Clean & Jerk results, which are stored in the database. The physical attributes of height and weight are also recorded and used to enforce the team's physical criteria. The business configuration includes a maximum team size limit of 5 bodybuilders, a minimum average height requirement, and a maximum average weight requirement, all of which are critical to the selection process.
Goals
The primary goal of this optimization problem is to maximize the total performance score of the selected team. This score is calculated as the sum of the Snatch and Clean & Jerk scores of the chosen bodybuilders. Success is measured by achieving the highest possible total score while adhering to the constraints on team size, average height, and average weight.
Constraints
The selection of bodybuilders for the team must respect the following constraints: 1. **Team Size Limit**: The total number of bodybuilders selected for the team must not exceed the predefined limit of 5. This ensures the team remains manageable and diverse. 2. **Minimum Average Height**: The average height of the selected bodybuilders must be at least 170 cm. This ensures the team meets the competition's physical standards for height. 3. **Maximum Average Weight**: The average weight of the selected bodybuilders must not exceed 100 kg. This ensures the team aligns with the competition's physical standards for weight.
These constraints are designed to ensure the team is both competitive and compliant with the competition's requirements.

Available Data - Database Schema

```
CREATE TABLE body_builder (
  Snatch FLOAT,
  Clean_Jerk FLOAT
);

CREATE TABLE people (
  Height FLOAT,
  Weight FLOAT
);

CREATE TABLE team_selection (
  is_selected BOOLEAN
);

INSERT INTO body_builder (Snatch, Clean_Jerk) VALUES (150.5, 200.0);
INSERT INTO body_builder (Snatch, Clean_Jerk) VALUES (160.0, 210.5);
INSERT INTO body_builder (Snatch, Clean_Jerk) VALUES (170.5, 220.0);

INSERT INTO people (Height, Weight) VALUES (175.0, 90.0);
INSERT INTO people (Height, Weight) VALUES (180.0, 95.0);
INSERT INTO people (Height, Weight) VALUES (185.0, 100.0);

INSERT INTO team_selection (is_selected) VALUES (True);
INSERT INTO team_selection (is_selected) VALUES (False);
INSERT INTO team_selection (is_selected) VALUES (True);
```

Data Dictionary - **body_builder Table**: - **Snatch**: The Snatch score of a bodybuilder, used to calculate the total performance score. - **Clean_Jerk**: The Clean & Jerk score of a bodybuilder, used to calculate the total performance score. - **people Table**: - **Height**: The height of a bodybuilder in centimeters, used to enforce the minimum average height constraint. - **Weight**: The weight of a bodybuilder in kilograms, used to enforce the maximum average weight constraint. - **team_selection Table**: - **is_selected**: A binary indicator of whether a bodybuilder is selected for the team, representing the decision variable in the optimization model.

Mathematical Optimization Formulation

Decision Variables - Let x_i be a binary decision variable where: - $x_i = 1$ if bodybuilder i is selected for the team. - $x_i = 0$ otherwise.
Here, $i = 1, 2, 3$ corresponds to the three bodybuilders in the database.
Objective Function
Maximize the total performance score:

$$\text{Maximize } Z = 350.5x_1 + 370.5x_2 + 390.5x_3$$

Constraints

1. Team Size Limit: $x_1 + x_2 + x_3 \leq 5$ (15)
2. Minimum Average Height: $5.0x_1 + 10.0x_2 + 15.0x_3 \geq 0$ (16)
3. Maximum Average Weight: $-10.0x_1 - 5.0x_2 + 0.0x_3 \leq 0$ (17)

Cross-Solver Analysis and Final Recommendation

Solver Results Comparison

Solver	Status	Optimal Value	Execution Time
Gurobipy	OPTIMAL	1111.50	0.17s
Docplex	OPTIMAL	1111.50	1.06s
Pyomo	OPTIMAL	1111.50	0.78s

Solver Consistency Analysis
Result: All solvers produced consistent results ✓
Consistent Solvers: gurobipy, docplex, pyomo
Majority Vote Optimal Value: 1111.5
Final Recommendation
Recommended Optimal Value: 1111.5
Confidence Level: HIGH
Preferred Solver(s): gurobipy
Reasoning: Gurobipy is recommended due to its fastest execution time while achieving the same optimal solution as the other solvers.

C.4 Example 4: Bodybuilder Performance Optimization (GPT-4o-Generated)

Building on similar data structures, this variant explores training optimization where impact coefficients modify the relationship between different exercises and overall performance targets.

Problem Context, Goals and Constraints

Context
The fitness organization is focused on enhancing the competitive performance of bodybuilders by optimizing their training regimen. The primary decision variables in this optimization are the weights lifted in the Snatch and Clean & Jerk events. These variables are continuous and directly mapped to the bodybuilder's performance in these lifts. The operational goal is to maximize the total weight lifted across these events, aligning with the linear objective of summing the weights lifted in Snatch and Clean & Jerk.
The business configuration includes several key parameters: the target total weight to be lifted by a bodybuilder, which serves as a constraint in the optimization model, and the impact coefficients for Snatch and Clean & Jerk training, which adjust the focus of training in the optimization model. These parameters are crucial for ensuring that the optimization aligns with realistic training impacts and performance targets.
The organization uses current operational data to inform decision-making, ensuring that the optimization problem remains grounded in practical, achievable goals. The constraints are designed to reflect resource limitations and performance targets, ensuring that the optimization remains linear and avoids nonlinear relationships such as variable products or divisions. The business configuration parameters are referenced throughout to maintain consistency and alignment with the optimization objectives.
Goals
The primary goal of the optimization is to maximize the total weight lifted by bodybuilders in competitions. This is achieved by focusing on the Snatch and Clean & Jerk lifts, with the objective being to maximize the sum of the weights lifted in these events. Success is measured by the alignment of the optimization with the expected impact coefficients for training, ensuring that the focus on Snatch and Clean & Jerk lifts leads to improved competitive performance. The optimization goal is clearly defined in natural language, emphasizing the linear nature of the objective without resorting to mathematical formulas or symbolic notation.
Constraints
The optimization problem includes constraints that ensure the total weight lifted by each bodybuilder does not exceed specified limits. These constraints are directly mapped to the total weight lifted by the bodybuilder, ensuring that the optimization remains within realistic performance boundaries. Additionally, each bodybuilder has a performance target, which serves as a constraint in the optimization model. These constraints are described in business terms, naturally leading to linear mathematical forms without involving variable products or divisions.

Available Data - Database Schema

```
CREATE TABLE body_builder (
    Snatch FLOAT,
    Clean_Jerk FLOAT,
    Total FLOAT,
    Snatch_Impact FLOAT,
    Clean_Jerk_Impact FLOAT
);

CREATE TABLE bodybuilder_performance (
    Bodybuilder_ID INTEGER,
    Performance_Target FLOAT
);

INSERT INTO body_builder VALUES
    (85.0, 105.0, 190.0, 1.2, 1.5),
    (95.0, 115.0, 210.0, 1.3, 1.6),
    (100.0, 120.0, 220.0, 1.1, 1.4);

INSERT INTO bodybuilder_performance VALUES
```

```
(1, 300.0), (2, 320.0), (3, 340.0);
```

Data Dictionary
The data dictionary provides a comprehensive mapping of tables and columns to their business purposes and optimization roles. The `body_builder` table stores individual lift data for bodybuilders, with columns representing the weight lifted in Snatch and Clean & Jerk, the total weight lifted, and the impact coefficients for training. These columns serve as decision variables and objective coefficients in the optimization model. The `bodybuilder_performance` table stores performance metrics and targets for each bodybuilder, linking performance data to individual bodybuilders and serving as a constraint in the optimization model.

Mathematical Optimization Formulation

Decision Variables - Let x_1 be the weight lifted in the Snatch event for a bodybuilder. - Let x_2 be the weight lifted in the Clean & Jerk event for a bodybuilder.
Objective Function
Maximize the total weight lifted:

$$\text{Maximize } Z = x_1 + x_2$$

Constraints
For each bodybuilder, we have the following constraints:
1. **Performance Target Constraint**: The total weight lifted should not exceed the performance target for each bodybuilder. - For Bodybuilder 1: $x_1 + x_2 \leq 300.0$ - For Bodybuilder 2: $x_1 + x_2 \leq 320.0$ - For Bodybuilder 3: $x_1 + x_2 \leq 340.0$
2. **Training Impact Constraints**: These constraints ensure that the training impact coefficients are considered in the optimization. - For Bodybuilder 1: $1.2 \times x_1 + 1.5 \times x_2 \leq 190.0$ - For Bodybuilder 2: $1.3 \times x_1 + 1.6 \times x_2 \leq 210.0$ - For Bodybuilder 3: $1.1 \times x_1 + 1.4 \times x_2 \leq 220.0$
3. **Non-negativity Constraints**: The weights lifted must be non-negative. $x_1 \geq 0, x_2 \geq 0$

Cross-Solver Analysis and Final Recommendation

Solver Results Comparison

Solver	Status	Optimal Value	Execution Time
Gurobipy	OPTIMAL	519.87	1.21s
Docplex	OPTIMAL	519.87	6.89s
Pyomo	OPTIMAL	519.87	6.11s

Solver Consistency Analysis
Result: All solvers produced consistent results ✓
Consistent Solvers: gurobipy, docplex, pyomo
Majority Vote Optimal Value: 519.8717948717948
Final Recommendation
Recommended Optimal Value: 519.8717948717948
Confidence Level: HIGH
Preferred Solver(s): gurobipy
Reasoning: Gurobipy is recommended due to its faster execution time and precise results, making it suitable for time-sensitive applications.

The final two examples explore resource scheduling in the entertainment industry, focusing on cinema operations. These demonstrate more complex constraint structures involving capacity limitations, temporal scheduling, and revenue maximization across multiple venues, highlighting the framework's ability to discover sophisticated optimization opportunities in service operations.

C.5 Example 5: Cinema Scheduling Optimization (DeepSeek-V3-Generated)

This scheduling problem demonstrates how temporal constraints and capacity limitations combine with pricing structures to form a revenue maximization model, validated by `Gurobipy` and `DOCplex` consensus.

Problem Context, Goals and Constraints

1. Problem Context and Goals
Context
A cinema chain is focused on maximizing its revenue by optimizing the scheduling of films across its cinemas. The key decision involves determining the number of showings per film per cinema per day, which directly impacts revenue. The cinema operates under specific operational parameters, including the price per showing, the capacity of each cinema, and the maximum number of showings allowed per day per cinema. These parameters are critical in ensuring that the scheduling aligns with both business goals and operational constraints.
The business configuration includes two key scalar parameters: 1. **Maximum number of showings allowed per day per cinema**: This parameter ensures that the total number of showings per day does not exceed a realistic limit, which is set to 12 based on typical cinema operating hours. 2. **Total capacity of the cinema per day**: This parameter ensures that the total number of seats

available across all showings in a day does not exceed the cinema's daily capacity, which is calculated based on the cinema's seating capacity and the maximum number of showings.
The optimization problem is designed to maximize revenue by leveraging these parameters in a linear manner, ensuring that the relationships between decision variables and constraints remain straightforward and avoid any nonlinear complexities.
Goals
The primary goal of this optimization problem is to maximize the total revenue generated from film showings across all cinemas. Revenue is calculated by multiplying the price per showing, the number of showings per film per cinema per day, and the capacity of the cinema. Success is measured by achieving the highest possible revenue while adhering to the operational constraints, such as the maximum number of showings and the total capacity of the cinema. The optimization process ensures that these goals are met through a linear formulation, avoiding any nonlinear relationships that could complicate the decision-making process.
Constraints
The optimization problem is subject to the following constraints, which are designed to reflect realistic operational limitations: 1. **Maximum showings per day per cinema**: The total number of showings per day in a cinema cannot exceed the maximum allowed, which is set to 12. This ensures that the cinema's operating hours are not overextended. 2. **Total capacity per day**: The total number of seats available across all showings in a day must not exceed the cinema's daily capacity. This ensures that the cinema does not overbook its available seating. 3. **Minimum showings per film**: Each film must be shown at least once per day in each cinema. This ensures that all films receive adequate exposure and that the cinema's schedule remains balanced.

Available Data - Database Schema

```
CREATE TABLE schedule (
  Price FLOAT,
  Show_times_per_day INTEGER
);

CREATE TABLE cinema (
  Capacity INTEGER
);

INSERT INTO schedule (Price, Show_times_per_day) VALUES
  (12.99, 3);
INSERT INTO schedule (Price, Show_times_per_day) VALUES
  (9.99, 2);
INSERT INTO schedule (Price, Show_times_per_day) VALUES
  (7.99, 1);

INSERT INTO cinema (Capacity) VALUES (150);
INSERT INTO cinema (Capacity) VALUES (200);
INSERT INTO cinema (Capacity) VALUES (100);
```

Data Dictionary
The data dictionary provides a clear mapping of the tables and columns to their business purposes and optimization roles: - **schedule**: This table stores information about film showings, including the price per showing and the number of showings per film per cinema per day. - **Price**: Represents the price per showing of a film. This value is used as a coefficient in the revenue calculation. - **Show_times_per_day**: Represents the number of showings per film per cinema per day. This is the primary decision variable in the optimization problem. - **cinema**: This table stores information about cinemas, including their seating capacity. - **Capacity**: Represents the seating capacity of the cinema. This value is used as a coefficient in the revenue calculation.

Mathematical Optimization Formulation

Decision Variables
Let $x_{f,c}$ be the number of showings per day for film f in cinema c.
This is the primary decision variable, representing the number of showings per film per cinema per day.
Objective Function
Maximize the total revenue:

$$\text{Maximize} \sum_{f} \sum_{c} (\text{Price}_f \times \text{Capacity}_c \times x_{f,c})$$

Where: - Price_f is the price per showing for film f (from `schedule.Price`). - Capacity_c is the seating capacity of cinema c (from `cinema.Capacity`). - $x_{f,c}$ is the number of showings per day for film f in cinema c.
Constraints

1. Maximum showings per day per cinema: (18)

$$\sum_{f} x_{f,c} \leq 12 \quad \forall c \tag{19}$$

This ensures the total number of showings per day in each cinema does not exceed 12. (20)

2. Total capacity per day: (21)

$$\sum_{f} (\text{Capacity}_c \times x_{f,c}) \leq \text{Capacity}_c \times 12 \quad \forall c \tag{22}$$

This ensures the total number of seats available across all showings in a day does not exceed the cinema's daily capacity. (23)

3. Minimum showings per film: (24)

$$x_{f,c} \geq 1 \quad \forall f, c \tag{25}$$

This ensures each film is shown at least once per day in each cinema. (26)

Cross Solver Analysis and Final Recommendation

Solver Results Comparison

Solver	Status	Optimal Value	Execution Time
Gurobipy	OPTIMAL	66546.00	0.20s
Docplex	OPTIMAL	66546.00	1.44s
Pyomo	ERROR	N/A	0.74s

Solver Consistency Analysis
Result: All solvers produced consistent results
Consistent Solvers: gurobipy, docplex
Majority Vote Optimal Value: 66546.0
Final Recommendation
Recommended Optimal Value: 66546.0
Confidence Level: HIGH
Preferred Solver(s): gurobipy
Reasoning: Gurobipy is recommended due to its optimal solution, high reliability, and fastest execution time. DOCplex also found the same optimal solution but was less efficient. Pyomo is not recommended due to its execution error.

C.6 Example 6: Cinema Scheduling Optimization (GPT-4o-Generated)

A variant scheduling formulation that explores different constraint structures for the same business domain, showing how our framework can generate diverse valid formulations from similar operational contexts.

Problem Context, Goals and Constraints

Context
The cinema chain is focused on maximizing its revenue by strategically scheduling films across its various locations. Each cinema has a specific capacity and a limited number of show times available per day. The decision-making process involves determining the number of times each film should be shown in each cinema. This decision is represented by integer variables, where each variable corresponds to the number of screenings for a particular film in a specific cinema. The primary objective is to maximize the total revenue generated from these screenings. This is achieved by considering the price per screening for each film in each cinema, which serves as the coefficient in the revenue calculation. The operational parameters include the maximum number of screenings allowed per day for each cinema and the seating capacity, which must not be exceeded. These constraints ensure that the scheduling decisions remain feasible and align with the cinema's operational capabilities.
Goals
The primary goal of this optimization problem is to maximize the total revenue from film screenings across all cinemas. The metric for optimization is the total revenue, which is calculated by summing the product of the number of screenings and the price per screening for each film in each cinema. Success in this context is measured by the ability to achieve the highest possible revenue while adhering to the operational constraints of each cinema. The optimization goal is clearly defined in linear terms, focusing on maximizing revenue without involving complex mathematical operations.
Constraints
The scheduling decisions are subject to several constraints that ensure the feasibility of the solution: - Each cinema has a maximum number of screenings it can accommodate per day. The total number of screenings scheduled in a cinema must not exceed this limit. - The number of attendees for each screening, based on average attendance, must not exceed the seating capacity of the cinema. This ensures that the cinema does not overbook and maintains a comfortable viewing experience for patrons.

Available Data - Database Schema

```
CREATE TABLE cinema (
  Cinema_ID INTEGER,
  Capacity INTEGER,
  Max_Screenings_Per_Day INTEGER
);

CREATE TABLE film_schedule (
  Cinema_ID_Film_ID INTEGER,
  Show_Times INTEGER
);

CREATE TABLE film_pricing (
  Cinema_ID_Film_ID INTEGER,
  Price FLOAT
);

INSERT INTO cinema VALUES
  (1, 120, 5), (2, 180, 6), (3, 250, 7);

INSERT INTO film_schedule VALUES
  (101, 3), (102, 4), (103, 2);

INSERT INTO film_pricing VALUES
```

```
(101, 12.0), (102, 15.0), (103, 10.0);
```

Data Dictionary
The data dictionary provides a business-oriented view of the tables and columns, highlighting their roles in the optimization process: - **Cinema Table**: This table contains information about each cinema, including its unique identifier, seating capacity, and the maximum number of screenings allowed per day. These attributes are crucial for defining the constraints related to capacity and scheduling limits. - **Film Schedule Table**: This table records the number of times each film is shown in each cinema. The entries in this table represent the decision variables in the optimization problem, determining the scheduling strategy for maximizing revenue. - **Film Pricing Table**: This table provides the pricing information for each film in each cinema. The price per screening serves as the coefficient in the revenue calculation, directly influencing the optimization objective.

Mathematical Optimization Formulation

Decision Variables
- Let x_{ij} be the number of screenings for film j in cinema i. - x_{ij} is an integer variable representing the decision of how many times film j is shown in cinema i.
Objective Function
Maximize the total revenue from all screenings across all cinemas:

$$\text{Maximize } Z = \sum_{i,j} \text{Price}_{ij} \times x_{ij}$$

where Price_{ij} is the price per screening for film j in cinema i.
Constraints

1. Maximum Screenings per Cinema: (27)

For each cinema i, the total number of screenings must not exceed the maximum allowed: (28)

$$\sum_{j} x_{ij} \leq \text{Max\_Screenings\_Per\_Day}_{i} \quad (29)$$

2. Seating Capacity Constraint: (30)

For each cinema i and film j, the expected number of attendees per screening must not exceed the cinema's capacity. (31)

Assuming average attendance per screening is a known parameter $\text{Avg\_Attendance}_{ij}$: (32)

$$\text{Avg\_Attendance}_{ij} \times x_{ij} \leq \text{Capacity}_{i} \quad (33)$$

3. Non-negativity and Integer Constraints: (34)

$x_{ij} \geq 0$ and integer for all i, j. (35)

Data Source Verification:
- Price_{ij} comes from the `film_pricing.Price` column. - $\text{Max\_Screenings\_Per\_Day}_{i}$ comes from the `cinema.Max_Screenings_Per_Day` column. - Capacity_{i} comes from the `cinema.Capacity` column. - $\text{Avg\_Attendance}_{ij}$ is assumed to be a known parameter from business configuration or historical data.
This linear programming model is designed to maximize the total revenue from film screenings while adhering to the operational constraints of each cinema. The decision variables, objective function, and constraints are all expressed in linear terms, ensuring the model is suitable for linear or mixed-integer programming solvers.

Cross-Solver Analysis and Final Recommendation

Solver Results Comparison

Solver	Status	Optimal Value	Execution Time
Gurobipy	OPTIMAL	88.00	0.51s
Docplex	OPTIMAL	12.00	7.74s
Pyomo	OPTIMAL	88.00	4.29s

Solver Consistency Analysis
Result: Solvers produced inconsistent results
Consistent Solvers: gurobipy, pyomo
Inconsistent Solvers: docplex
Potential Issues: - DOCplex may have encountered a different local optimum or misinterpreted constraints. - Possible data input errors or solver configuration issues specific to DOCplex.
Majority Vote Optimal Value: 88.0
Final Recommendation
Recommended Optimal Value: 88.0
Confidence Level: HIGH
Preferred Solver(s): multiple
Reasoning: Both Gurobipy and Pyomo provided consistent and high objective values, indicating a reliable solution. Using multiple solvers can validate results and ensure robustness.

These examples collectively shows that our bottom-up approach successfully discovers a wide range of optimization patterns directly from database structures. The consistent achievement of solver consensus across diverse problem types validates both our generation methodology and the quality of

the resulting benchmark. The natural emergence of linear programming formulations from relational data structures suggests that many real-world business databases inherently encode optimization opportunities waiting to be discovered through systematic analysis.

D ALGORITHMIC DETAILS OF SCHEMA2OPT GENERATION

The Schema2Opt generation framework employs an alternating optimization approach that iteratively refines both database schemas and optimization formulations through structured dialogue between specialized agents. This section provides the core algorithmic specifications and key prompt engineering strategies that enable automatic discovery of optimization opportunities from database structures.

D.1 ALTERNATING OPTIMIZATION ALGORITHM

Our generation process alternates between two types of analysis until convergence. First, an OR Expert examines the current database schema to identify what optimization problem could be solved with the available data. Then, a Data Engineer modifies the schema to better support the optimization requirements identified by the OR Expert. This back-and-forth continues until the schema contains all necessary information to formulate a complete optimization problem.

Algorithm 1 Schema2Opt Alternating Generation

Input: Spider database schema with at most 5 tables
Output: Business document, database content, verified optimal value
Parameters: Maximum 5 iterations, convergence threshold 0.99
// Phase 1: Iterative schema refinement
Initialize with original database schema
OR Expert analyzes schema to design optimization problem
while not converged and iterations ¡ 5 **do**
  // Data Engineer modifies schema based on OR analysis
  Identify missing data elements from OR Expert feedback
  Create new tables or columns for decision variables
  Add fields for objective coefficients and constraints
  Move single-value parameters to configuration file
  // OR Expert evaluates improved schema
  Check if all optimization components have data sources
  Calculate mapping adequacy score (0 to 1)
  Mark complete if adequacy exceeds 0.99
end while
// Phase 2: Generate realistic business scenario
Triple Expert creates realistic data values for all tables
Generate natural language business description
Extract mathematical formulation from business context
// Phase 3: Validate with multiple solvers
Generate code for Gurobi, DOCplex, and Pyomo
Execute each solver and collect optimal values
Accept if at least 2 solvers agree on the solution
Otherwise discard as invalid instance

The convergence process typically completes within three to four iterations. The mapping adequacy score combines assessments of three key components: whether objective function coefficients can be found in the data (weighted at 35 percent), whether decision variables are properly represented (35 percent), and whether constraint parameters are available (30 percent). When this combined score exceeds 99 percent, we consider the schema sufficiently complete for optimization.

D.2 KEY PROMPT ENGINEERING STRATEGIES

The success of Schema2Opt generation relies heavily on carefully designed prompts that enforce agent specialization and maintain mathematical rigor. We present the core strategies that enable effective agent collaboration.

D.2.1 OR EXPERT PROMPT DESIGN

The OR Expert prompt enforces strict linearity constraints while discovering optimization opportunities from database structures. The prompt explicitly prohibits nonlinear operations and guides the agent to identify how business scenarios naturally map to linear formulations. A critical innovation is the row count awareness mechanism, where the agent understands that tables requiring fewer than 3 meaningful rows will be moved to configuration files, influencing parameter mapping decisions.

OR Expert Core Instructions:

CRITICAL MATHEMATICAL CONSTRAINTS:

- The optimization MUST be Linear Programming (LP) or Mixed-Integer (MIP)
- Objective: minimize/maximize $\sum$(coefficient $\times$ variable) ONLY
- Constraints: $\sum$(coefficient $\times$ variable) $\leq$/$\geq$/= constant
- NO variable products (x$\times$y), divisions (x/y), or nonlinear terms
- Generate between 3-10 constraints for optimization feasibility

MAPPING EVALUATION:
For each optimization component, assess mapping adequacy [0.0-1.0]:

- 1.0: Directly available in current schema
- $0.5 - 0.9$: Can be derived from existing data
- < 0.5: Missing, requires schema modification

D.2.2 DATA ENGINEER PROMPT DESIGN

The Data Engineer prompt implements schema modifications while maintaining database normalization principles. A key design principle embedded in the prompt is the distinction between tabular and scalar data. The prompt instructs that collections naturally forming multiple rows belong in database tables, while single-value parameters belong in configuration files. This separation ensures proper data organization and prevents the creation of single-row tables that would complicate the optimization model.

Data Engineer Core Instructions:

DATA ORGANIZATION PRINCIPLES:

- TABULAR DATA (database): Collections of similar items
  - Multiple products, locations, time periods
  - Individual costs, demands per entity
  - Many-to-many relationships
- SCALAR PARAMETERS (config): Single business values
  - Global limits: "daily_capacity": 1000
  - Thresholds: "min_stock_ratio": 0.2
  - Business rules: "max_activities": 2

IMPLEMENTATION RULES:
Apply 3-row minimum: If optimization data cannot generate $\geq$3 meaningful rows, move to configuration instead of creating sparse tables.

D.2.3 TRIPLE EXPERT DATA GENERATION

The Triple Expert combines three domains of expertise to generate realistic data that ensures optimization solvability. The prompt instructs the agent to consider business norms, maintain cross-table consistency, and create meaningful trade-offs in the optimization problem. Values must reflect industry standards while ensuring the problem remains neither trivial nor infeasible.

Triple Expert Core Instructions:

EXPERTISE SYNTHESIS:

- Business Operations: Values reflect industry norms and realistic scenarios
- Data Management: Maintain referential integrity and cross-table consistency

- Optimization Modeling: Ensure feasible solutions with meaningful trade-offs

DATA GENERATION CONSTRAINTS:

- Generate 3-100 rows per table based on business context
- Coefficients create non-trivial optimization (avoid dominant solutions)
- Constraint bounds allow feasible region but force trade-offs
- Configuration parameters use realistic business values

E IMPLEMENTATION DETAILS OF DATA2DECISION

Following the Schema2Opt generation framework, we present the algorithmic details and implementation strategies of our Data2Decision prescriptive analytics data agent. While Schema2Opt creates the benchmark through alternating optimization between agents, Data2Decision solves these problems through parallel test-time scaling with multi-solver consensus.

E.1 DATA2DECISION ALGORITHM

The Data2Decision system processes each prescriptive analytics problem through multiple parallel attempts, each exploring different interpretations of the data-to-optimization mapping. The algorithm orchestrates these attempts and aggregates results through majority voting to produce robust solutions.

Algorithm 2 Data2Decision Parallel Solving with Consensus

Input: Business document, database schema, database content
Output: Optimal value with confidence score
Parameters: Number of attempts $N = 10$
// Execute N parallel solution attempts
Initialize empty result set for storing successful attempts
for each attempt i from 1 to N in parallel **do**
  // Stage 1: Extract data from database
  Set SQL temperature based on attempt number
  Generate SQL queries to extract optimization parameters
  Execute queries and create enhanced problem description
  // Stage 2: Generate and execute solver code
  Set code temperature based on attempt number
  Select solver cyclically (Gurobi, DOCplex, Pyomo)
  Generate solver-specific optimization code
  Execute solver with 300-second timeout
  Extract optimal value from solver output
  if execution successful **then**
    Add optimal value to result set
  end if
end for
// Aggregate results through majority voting
if no successful attempts **then**
  return failure
end if
Count frequency of each unique optimal value
Select most frequent value as final result
Calculate confidence as fraction of attempts agreeing
return optimal value and confidence score

The algorithm achieves robustness through systematic exploration of the solution space. By varying temperatures and rotating solvers across attempts, we capture diverse valid interpretations of the prescriptive analytics problem. The majority voting mechanism identifies the most reliable solution, with the consensus strength providing a confidence measure. Temperature ranges from 0.05 to 0.50 for SQL generation and 0.00 to 0.18 for code generation, with linear increments across attempts.

E.2 SQL Query Generation Strategy

The SQL generation process in Stage 1 employs a structured prompting approach that guides the language model to identify relevant data for optimization. Unlike traditional Text-to-SQL systems that answer specific questions, our approach requires the model to proactively discover what data elements map to optimization components.

The prompt structure instructs the model to analyze the problem through an optimization lens, identifying data needed for decision variables (what to optimize), objective coefficients (what to maximize or minimize), and constraint parameters (limitations and requirements). The temperature-controlled generation explores different interpretations of these mappings, particularly valuable when business requirements use ambiguous terminology or implicit relationships.

We implement a query extraction mechanism that parses the LLM response to identify valid SQL statements. The system handles both structured responses with SQL code blocks and unstructured responses where queries are embedded in natural language explanations. Each extracted query is validated for syntactic correctness before execution against an in-memory SQLite database created from the provided schema and data files.

E.3 Direct Code Generation Strategy

Stage 2 implements direct optimization code generation without intermediate mathematical modeling. This design choice, validated through our ablation studies, reduces error propagation and allows the language model to leverage its internalized optimization knowledge more effectively.

The solver-specific code generation uses template-guided prompting that provides concise implementation patterns for each framework. For Gurobi, we emphasize the use of quicksum for efficient constraint aggregation and proper variable bounds specification. For DOCplex, we highlight the framework's functional API and constraint naming conventions. For Pyomo, we demonstrate the rule-based constraint definition pattern that differs from imperative approaches.