Questions Are All You Need: Parallel Agents for Data Visualization

Ji Hyung Kim*

Microsoft

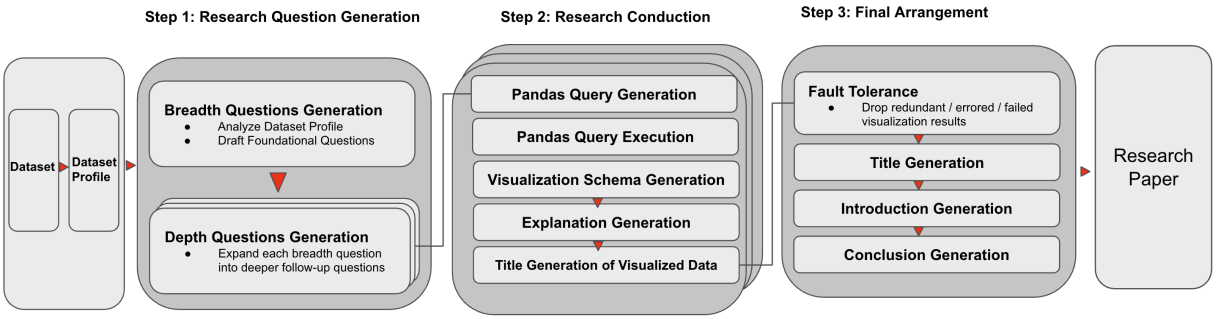

Figure 1: Question-driven parallel agent architecture with three-level parallelization hierarchy: (1) depth question generation from breadth questions, and (2) concurrent research execution across all questions with nested component-level parallelism.

Abstract

This paper presents a lightweight framework for automated data analysis using large language models (LLMs). Rather than relying on manual feature engineering, our system begins with LLM-generated research questions derived from dataset statistical profiles. These questions then guide computation, visualization, and narrative synthesis, ensuring that analysis is driven by inquiry rather than column selection. The architecture employs three levels of parallelization: depth question expansion, question-level research execution, and component-level processing within individual questions. Each question functions as an independent analytical unit, enabling concurrency through type-safe abstractions, systematic caching, and isolated fault handling. By inverting the conventional pipeline, the system avoids cascading errors from feature selection, supports scalability across diverse datasets, and achieves consistent analytical coverage. Our results demonstrate that robust automated analysis is less a matter of complex downstream modeling than of rigorous upstream problem formulation. We argue that centering analysis on research questions provides a practical path toward reliable, parallel, and extensible LLM-driven data visualization.

Index Terms: Agentic Systems, Automated Analysis, Data Visualization, Parallel Processing

1 Introduction

LLMs are increasingly applied to data analysis and visualization because of their ability to generate code, summarize findings, and adapt to diverse tasks. Yet their integration into structured data workflows exposes persistent challenges: token limitations, hallucinated fields, inconsistent formatting, and brittle dependency chains. Conventional pipelines that begin with feature engineering exacerbate these problems, as early mistakes propagate downstream through question generation, computation, and visualization.

We take a different perspective: *questions are all you need*. Instead of beginning with features, our system generates research questions directly from dataset profiles. The dataset profile summarizes each column with key statistics—row counts, distinct and missing values, inferred data types, most frequent entries, and sample examples—providing a compact overview of the dataset's structure and content. These questions drive the entire workflow, from computation to visualization and reporting. By shifting the entry point from column selection to inquiry formulation, the system eliminates feature-selection bottlenecks, reduces cascading errors, and naturally adapts to datasets of varying structure.

The architecture consists of three stages with three levels of parallelization. Question generation agents create breadth questions sequentially, then expand them into depth questions in parallel. Research execution agents translate all questions into pandas code, execute computations, and generate visualizations concurrently, with nested parallelism for independent components within each question. Arrangement agents synthesize report elements (title, introduction, conclusion) sequentially while maintaining logical result organization. Because each question is independent, the system supports aggressive parallelization and robust fault isolation.

Our contributions are threefold:

1. Propose a question-first reformulation of automated data analysis, replacing feature engineering with inquiry-driven workflows.

2. Design a three-level parallel agent architecture with type-safe abstractions, caching, and error handling for scalable and reproducible execution.

3. Present design insights—from constraint-based prompt engineering to proactive filtering—that enable reliable LLM-based visualization pipelines.

By reframing the pipeline around questions, we show that effective automated analysis depends less on sophisticated downstream modeling and more on rigorous upstream problem formulation.

2 Challenges in LLM-Driven Data Analysis

Automated data analysis with LLMs exposes two classes of difficulties: inherent mismatches with structured datasets and practical implementation issues.

*e-mail: jihyungkim@microsoft.com

2.1 Inherent Challenges

- **Scale and structure**: Even moderately sized datasets exceed LLM token limits, forcing sampling strategies that risk losing critical patterns. Ambiguities in schema (categorical vs. continuous fields, multi-value columns, temporal variables) amplify misclassification errors and lead to invalid analyses.
- **Hallucination and reliability**: LLMs fabricate columns, produce malformed queries, or select inappropriate fields, creating downstream failures that undermine trust in results.

2.2 Development Challenges

- **Prompt and output control**: Despite detailed few-shot examples, models often overfit prompts, replicate irrelevant patterns, or return inconsistent formats (e.g., extra markdown, invalid code).
- **Question quality**: Generated queries can be redundant, computationally infeasible, or demand domain knowledge beyond dataset scope, requiring validation and fallback mechanisms.

3 Architecture Overview

3.1 Researcher System Design and Abstractions

Our system architecture centers around a dedicated `Researcher` class that encapsulates the entire analytical workflow through specialized data abstractions and configurable processing parameters. The `ResearchConfig` class provides fine-grained control over system behavior, including parallelization limits (`max_workers`), question generation parameters (`breadth` and `depth`), and performance optimizations (`use_caching`). This configuration-driven approach enables dynamic adaptation to computational constraints, API rate limits, and deployment environments while maintaining analytical consistency.

Critical to the system's modularity are custom data types that provide structured interfaces between processing stages. `ResearchQuestion` objects encapsulate question hierarchies with metadata including level (`breadth` vs. `depth`), `parent_question` relationships, suggested visualization types, analytical category, and relevant `source_columns`. `ResearchResult` objects aggregate computational outputs, storing `computed_data`, `visualization_code`, explanatory narratives, and generated titles as cohesive analytical units. These abstractions enable type-safe data flow between independent processing agents while facilitating caching, error handling, and result validation.

3.2 Three-Level Parallel Architecture

3.2.1 Level 1: Depth Question Expansion (Partial Parallelization)

The question generation stage employs targeted parallelization for depth question expansion. Breadth questions are generated sequentially through a single LLM call to ensure comprehensive dataset coverage. Each breadth question then spawns depth questions in parallel using `ThreadPoolExecutor`, where each breadth question receives dedicated thread resources for independent depth expansion. This approach balances systematic coverage with concurrent processing efficiency.

3.2.2 Level 2: Question-Level Research Execution (Full Parallelization)

The research execution stage implements comprehensive parallelization across all generated questions. Each `ResearchQuestion` object is processed independently through dedicated threads, with up to `max_workers` questions executing concurrently. This design eliminates sequential bottlenecks and enables linear scalability with question volume while maintaining fault isolation through independent thread execution.

3.2.3 Level 3: Component-Level Processing (Nested Parallelization)

Within individual question processing, visualization code generation and title generation execute concurrently as independent operations. This nested parallelism operates within each question thread from Level 2, utilizing a constrained thread pool (max 2 workers) to optimize resource utilization. Explanation generation remains sequential as it depends on completed visualization code, maintaining logical dependencies while maximizing concurrent processing.

4 Key Decisions

4.1 Architectural Evolution: From Feature Engineering to Question Generation

The most fundamental decision involved inverting the traditional data science pipeline from feature engineering first to question generation first. Our original approach followed conventional methodology: identify visualizable columns through LLM-driven feature engineering, then generate research questions from selected features. This created cascading dependencies where poor feature selection propagated through the entire pipeline, requiring complex error handling and recovery mechanisms.

The breakthrough insight—questions are all you need—led to a complete architectural inversion. The new approach generates research questions directly from dataset statistical profiles, then derives appropriate data computations and visualizations from those questions. This eliminates the feature selection bottleneck while naturally accommodating diverse dataset structures without predetermined assumptions about data relevance.

4.2 Hierarchical Parallelization Strategy and Fault Tolerance

Question-centric parallelization emerged as significantly more tractable than feature-centric parallelization. The three-level hierarchy enables targeted optimization: Level 1 balances coverage with concurrency, Level 2 maximizes question throughput, and Level 3 optimizes component generation. Each level maintains independent fault isolation, preventing error propagation while enabling partial result delivery.

The `ThreadPoolExecutor` implementation provides configurable concurrency control at each level, allowing optimization for diverse deployment scenarios while respecting API rate limits. Individual agent failures remain isolated within their respective threads, preventing system-wide cascades, while successful results aggregate seamlessly into comprehensive analytical outputs. This fault tolerance proved crucial given the inherent unreliability of LLM-generated content.

4.3 Computational Strategy: Pandas-First Execution

A critical decision involved eliminating LLM dependency during data computation phases. Rather than generating high-level analytical descriptions that require further LLM interpretation, the system generates executable pandas code directly. This ensures deterministic, fast computation with zero reliance on LLM availability or consistency during the execution phase.

The pandas-first approach provides additional benefits: computational results are immediately cacheable, errors are deterministic and debuggable, and performance scales with dataset size rather than LLM processing time. This decision proved essential for system reliability and development efficiency.

5 Conclusion

This work demonstrates that reframing automated analysis around questions rather than features yields both practical and conceptual benefits. By structuring the workflow as a parallel agent architecture, the system achieves reliable scalability across diverse datasets while

avoiding cascading failures common in feature-first pipelines. On the VisPub dataset, our current implementation achieved average execution times of 115 ± 5 seconds for typical research workloads, highlighting the efficiency of question-level parallelization in processing independent tasks concurrently while maintaining robustness through isolated fault handling.

Beyond performance, the question-first design improves analytical coverage and insight quality. The system can automatically identify relationships across heterogeneous columns without relying on predefined assumptions, thereby supporting richer and more adaptable inquiry. These results highlight a broader lesson: effective automated analysis depends less on complex downstream modeling and more on rigorous upstream formulation through intelligent question generation.

Looking forward, we envision extending this framework with adaptive question refinement, integration of domain knowledge for improved relevance, and broader benchmarking across datasets. Taken together, these directions position the question-first paradigm as a practical foundation for scalable, reliable, and extensible LLM-driven data visualization.