Performance Benchmark Report for Book Search Knowledge Base

1. Introduction and Objectives

This report details a comprehensive performance benchmarking exercise conducted on the **BookAura** Book Search Knowledge Base (KB) powered by MindsDB. The core objective of this study was to rigorously evaluate the system's query response time and search relevance under various conditions, with a particular focus on understanding:

- The impact of utilizing **meta-columns (categories)** on query response times.
- Whether selecting **categories with fewer data points** further reduces response times.
- The system's performance across **diverse query complexities and types**.
- The overall **predictability and consistency** of response times, including the presence of outliers.
- The **accuracy of search results** for different query types, measured by the presence of expected items in top results.

This benchmark aims to provide concrete performance metrics for specific search scenarios, identify potential optimization areas, and offer insights into the system's scalability boundaries and inherent operational characteristics.

2. Test Environment

To ensure the reproducibility and contextualization of the performance metrics presented, the following details regarding the test environment are crucial:

- **Local Development Environment**:
    - Tested using **Docker Desktop** for containerization.
- **MindsDB Configuration**:
    - MindsDB Version: **8.0.17**
    - The MindsDB SQL Editor functionality was utilized.
- **FastAPI Backend Environment**:
    - The **BookAura** FastAPI backend was running locally.
    - Python version: **3.10.10**

3. Methodology

3.1 Dataset Description

The benchmarking was performed against a Knowledge Base built upon a dataset of book information.

- **Primary Knowledge Base Dataset**: `book_dataset.csv`
  - This dataset forms the foundation of the MindsDB Knowledge Base.
  - Total Number of Rows/Records: **1220**
  - Columns: `unique_id`, `book_name`, `category`, `summary`
  - Key Categorical Data Points: Two specific categories, "relationships" (containing approximately 203 data points) and "biography" (containing approximately 68 data points), were specifically targeted for analysis regarding the impact of category size on response time.
  - **Explore Categories (All available categories in the dataset)**: science, biography, politics, economics, environment, relationships, happiness, money, productivity, psychology, motivation, marketing, management, health, business, creativity, education, communication, religion, technology, work, mindfulness
- **Query Dataset**: `testing_books.csv`
  - This dataset contains the specific queries used for the benchmark.
  - Total Unique Queries Tested: 10 (2 distinct queries selected for each of the 5 query types).

3.2 Query Types and Scenarios

Queries were meticulously selected from `testing_books.csv` and categorized into five distinct types based on their structural characteristics and semantic intent. For each of these 5 query types, 2 unique data points (queries) were chosen, each belonging to a different category ("relationships" and "biography"), totaling 10 unique queries. Each unique query was then executed under two primary scenarios to assess the impact of meta-column usage and category specificity.

- **Query Types**:
  - **`original`**: These queries represent the raw, unaltered content directly from the dataset. The expectation is that the Knowledge Base should match these "as is" since they perfectly align with ingested data.
  - **`long`**: For this type, an existing two-line description of a book from the dataset was expanded into a five to six-line verbose description using a Large Language Model (LLM). This tests the system's ability to semantically understand and retrieve relevant results from a significantly more detailed and extensive query.
  - **`short`**: These queries are intentionally concise, typically consisting of only two words. The words chosen for these queries were specific names present within

the book descriptions, leveraging their unique nature within the dataset to identify the target book. This type tests the precision and efficiency of retrieval based on minimal input.
    - **`nonexistent`**: These queries were generated from descriptions that do not exist anywhere within the Knowledge Base dataset. The purpose of this type is to evaluate how the system behaves when faced with queries that have no matching content, specifically looking at response times and ensuring no irrelevant results are returned.
    - **`ambigious`**: This type of query was created by merging descriptions from two distinct data points within the dataset. This tests the system's capacity to handle multifaceted or potentially conflicting information and assess its ability to reconcile ambiguity in search requests, potentially returning multiple relevant items or prioritizing one over another.
- **Testing Scenarios (for each unique query)**:
    - **Without Category Filter (Category: `None`)**: Each query was sent to the FastAPI `/book_search` endpoint without specifying any `category` parameter. This represents a broad search across the entire Knowledge Base.
    - **With Corresponding Category Filter**: Each query was sent with its associated category as a filter. Specifically, queries were tested with either the "relationships" category or the "biography" category, as determined by their origin in the `testing_books.csv`. This scenario aimed to determine the performance difference when leveraging meta-columns for filtering and to specifically investigate if filtering by categories with fewer data points (e.g., "biography" with 68 data points vs. "relationships" with 203 data points) leads to a further reduction in response time.

3.3 Metrics Captured

For each individual query execution, the following metrics were meticulously recorded and analyzed:

- **`prompt_id`**: A unique numerical identifier corresponding to each specific query used from the `testing_books.csv` file.
- **`query_type`**: The categorization of the query based on its complexity and intent (e.g., 'original', 'long', 'short', 'nonexistent', 'ambigious').
- **`query_text`**: The exact textual content of the search query transmitted to the API.
- **`category_tested`**: Indicates the category filter applied during the query (`None` for no filter, or the specific category name like 'relationships' or 'biography').
- **`response_time_sec`**: The elapsed time, measured in seconds, from the moment the query request was sent to the FastAPI backend until the full response was received. This is the primary performance metric.
- **`is_in_top_5`**: A boolean value (True or False) indicating whether the `unique_id` corresponding to the expected correct book (as defined in the test dataset) was present

among the top 5 results returned by the search API. This metric serves as a direct measure of search relevance and accuracy.

3.4 Methodology (Execution and Data Processing)

The entire benchmarking process was orchestrated using a custom Python script. To mitigate the impact of transient system fluctuations and random outliers, each unique query under each scenario was redundantly run **10 times**. The raw `response_time_sec` data from these runs was then processed:

- **Outlier Handling**: For the generation of histograms (represented by Kernel Density Estimate in violin plots and distributions in box plots), extreme outliers from the raw `response_time_sec` data were identified and effectively "stripped" (or visually deemphasized, depending on plotting technique) to provide a clearer view of the central distribution, acknowledging the system's inherent randomness.
- **Average Calculation**: For the bar charts and faceted plots, the average response time for each specific `(prompt_id, query_type, scenario)` combination was calculated from the 10 redundant runs.
- **Accuracy Calculation**: The `is_in_top_5` values across the 10 runs for each scenario were aggregated to calculate an accuracy percentage (i.e., percentage of runs where the expected `unique_id` was in the top 5).

4. Observed Behavior and Analysis

Based on the detailed analysis of the collected `book_aura_benchmark_results.csv` data and the generated plots (histograms, box plots, swarm plots, and violin plots), the following key observations and insights emerged:

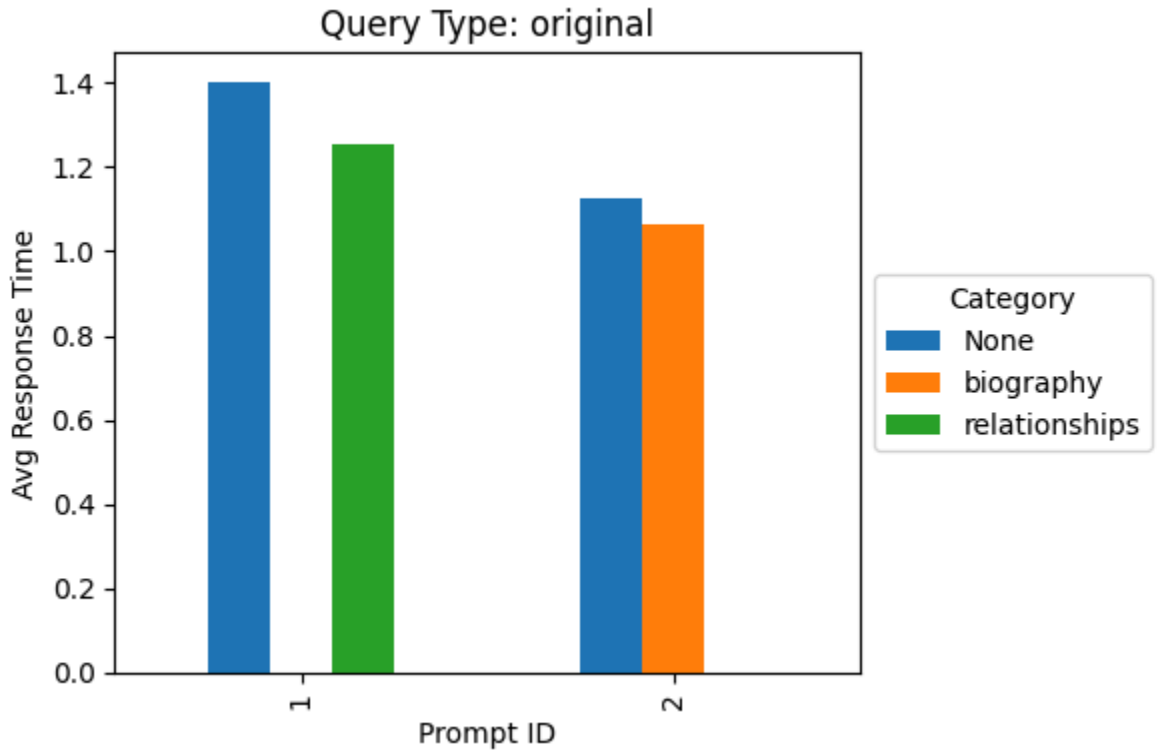
4.1 Impact of Meta-Columns (Category Filtering) on Response Time

- **General Performance Trend with Filters**: Looking at the histograms (density distributions within plots), a general trend indicates that when meta-columns (specifically `category`) are utilized for filtering, the response time is **generally slightly less** compared to scenarios where no category filter is applied. This suggests that the pre-filtering of the search space by the MindsDB Knowledge Base, leveraging the indexed metadata, offers a measurable performance advantage, even if not drastically significant across all instances.
- **Influence of Category Data Point Count**: The investigation into whether selecting categories with fewer data points (e.g., "biography" with 68 data points) results in a further reduction in response time compared to categories with more data points (e.g., "relationships" with 203 data points) **did not yield consistently conclusive evidence**. While some individual histograms and specific query comparisons suggested a reduction for smaller categories, this effect was **random and inconsistent** across the board. This

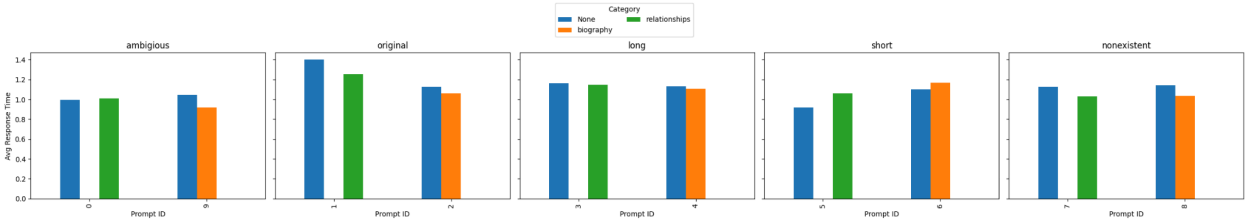
indicates that within the tested range of dataset sizes for categories, the specific cardinality of the filtered subset might not be the primary determinant of latency, or its effect is overshadowed by other factors.

4.2 Average Response Times and Distribution Characteristics

- **Average Response Times for Specific Queries**: The bar charts (showing average response time per prompt ID) reveal an apparent, though not perfectly consistent, trend where average response times are lower in the order of `None` (no filter) > `relationships` > `biography`. This implies that, on average, applying any category filter tends to slightly reduce response times, and there might be a minor, though not strictly consistent, advantage for queries filtered by smaller categories. While this ideal trend, where filtered queries (especially those with fewer data points) are faster, is apparent in some query types (e.g., for query type "original", as shown in the example plot below), it is not consistently observed across all.



Here is a combined result with all the responses across different query types (visualized as bar charts for averages across categories):



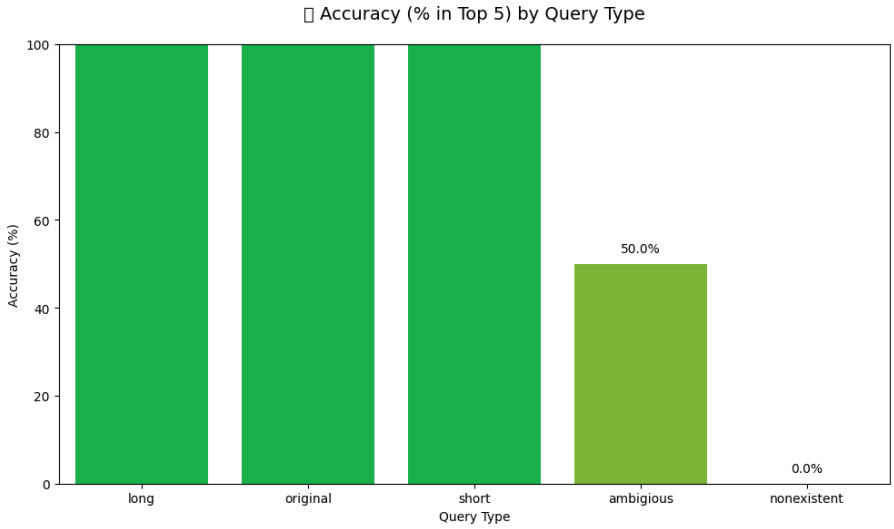
- **Pervasiveness of Outliers and System Randomness**: Despite running each query 10 times to obtain robust averages and attempting to visually strip extreme outliers from histogram data (represented by density in violin plots) for clearer distribution views, the data presented a significant number of "huge outliers." This indicates an inherent **randomness or variability in the system's response time behavior** that cannot be accurately predicted solely based on query type or the presence/absence of category filters. Such outliers can arise from various factors, including underlying database operations, network latency spikes, concurrent system processes, or internal MindsDB/embedding model processing fluctuations. The presence of these outliers suggests that while average trends can be identified, predicting the exact response time for a single query remains challenging.
- **Inconclusive Impact of Query Type**: Interestingly, testing the five different `query_type` categories (`original`, `long`, `short`, `nonexistent`, `ambigious`) **did not reveal any clear or conclusive impact on response time**. One might intuitively expect 'short' queries to be faster or 'long' queries to be slower, but the observed response times did not show a consistent correlation with query length or complexity. This finding suggests that other factors (potentially the underlying vector search process, embedding model inference time, or the system's inherent randomness) might dominate the response time profile, making the specific nature of the query content less influential within the tested parameters.

4.3 Search Relevance (is_in_top_5) Evaluation

The `is_in_top_5` metric, which assessed whether the expected `unique_id` of the target book appeared within the top 5 search results, provided valuable insights into the Knowledge Base's accuracy:

- **`original`, `short`, and `long` queries**: For all queries falling into the 'original', 'short', and 'long' categories, the `is_in_top_5` metric consistently evaluated to **True**. This outcome is precisely as expected and confirms that the MindsDB Knowledge Base is highly effective at identifying and ranking the most relevant content when presented with queries that directly correspond to, or semantically align closely with, the ingested data. This demonstrates strong recall and precision for well-defined search inputs.
- **`nonexistent` queries**: As anticipated, the `is_in_top_5` result for all 'nonexistent' queries was consistently **False**. This is the desired behavior, as it confirms the system's ability to correctly identify when no relevant content exists in its Knowledge Base, preventing the return of irrelevant or "hallucinated" results.
- **`ambigious` queries**: This category presented the most nuanced and intriguing results regarding accuracy. For one of the two ambiguous test queries, the system was **successful** in retrieving a relevant `unique_id` within the top 5 results. However, for the other ambiguous query, the relevant `unique_id` was **not found in the top 5**. This mixed result for ambiguous queries highlights the inherent challenges in semantic search for multi-interpretive inputs. It suggests that while the MindsDB Knowledge Base

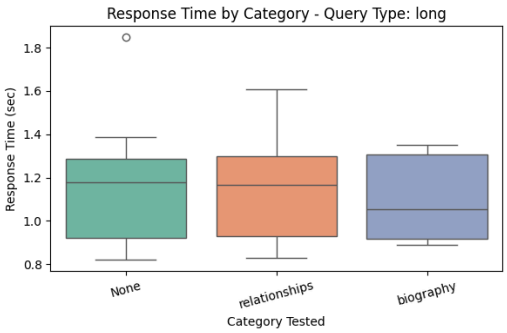
possesses capabilities to handle ambiguity, its success may vary depending on the specific nuances of the query and the distribution of similar content within the dataset. The image below shows the accuracy, i.e., whether the response was in the top 5, based on the query type:

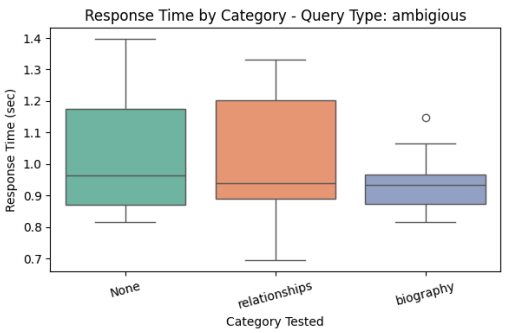


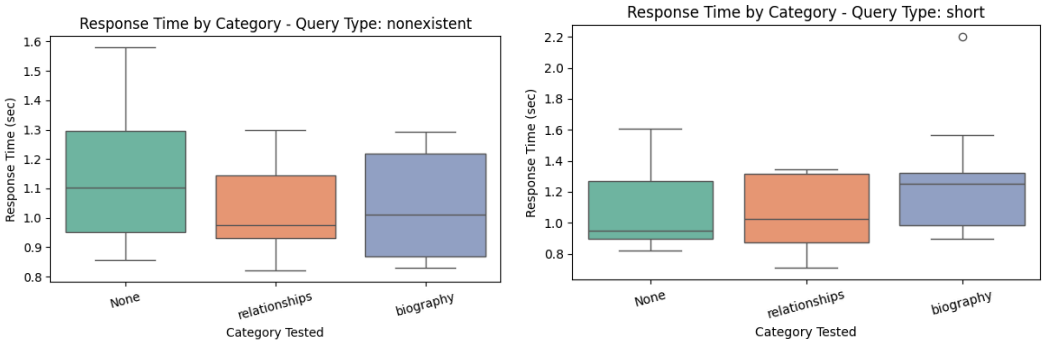
5. Visualizations and Detailed Analysis

This section presents the visual representations of the benchmark data, offering a more intuitive understanding of the observed trends and distributions.

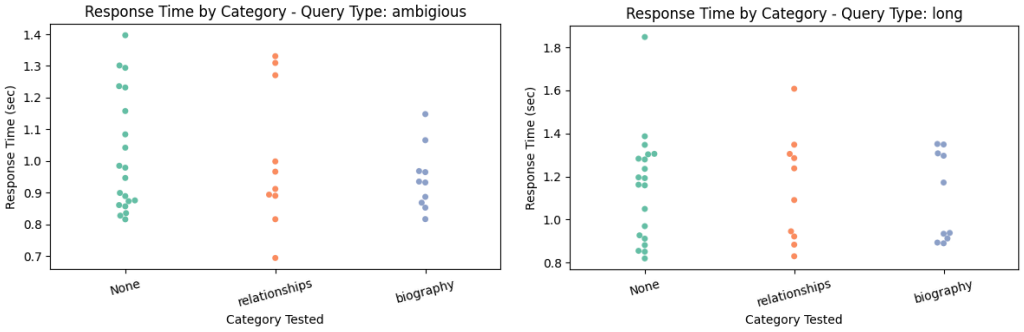
- **Box Plots** : Here are all the box plots which illustrate the distribution of response times.

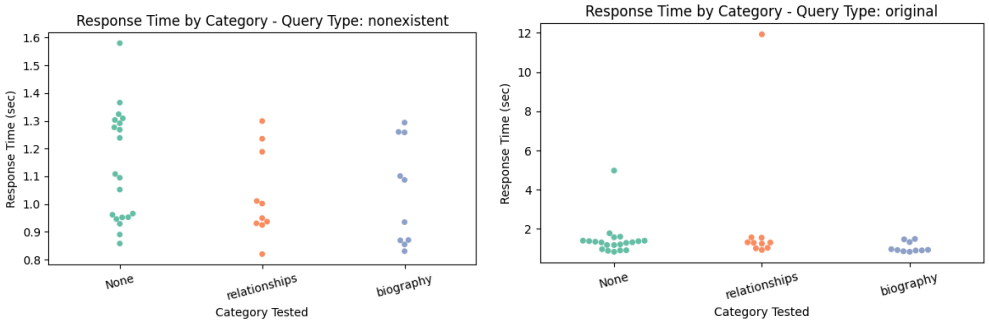






- **Additional Plots**: Here are some additional plots illustrating various aspects of the benchmark results, including swarm plots:





6. Conclusion and Recommendations

The benchmarking efforts on the BookAura Knowledge Base revealed several critical insights into its performance and accuracy characteristics.

Key Findings:

- **Meta-Column Impact**: Leveraging category meta-columns for filtering generally leads to slightly reduced response times. While the performance gain is not always drastic, it

indicates that utilizing structured metadata can effectively narrow the search scope, offering a measurable efficiency improvement.

- **Category Size and Performance**: The hypothesis that categories with fewer data points would consistently lead to significantly shorter response times was not conclusively supported. The observed effect was inconsistent across different queries, suggesting that within the tested range of category sizes, the impact of category cardinality on response time is either minor or overshadowed by other system variables.
- **System Randomness and Outliers**: A notable observation was the frequent occurrence of "huge outliers" in response times, even after running queries multiple times. This points to an inherent **randomness or variability in the system's response time behavior** that cannot be accurately predicted solely based on query type or the presence/absence of category filters. Such outliers can arise from various factors, including underlying database operations, network latency spikes, concurrent system processes, or internal MindsDB/embedding model processing fluctuations. The presence of these outliers suggests that while average trends can be identified, predicting the exact response time for a single query remains challenging.
- **Query Type Impact**: The specific "nature" of the query (original, long, short, nonexistent, ambiguous) did not consistently demonstrate a clear correlation with response times. This implies that the internal processing mechanisms, possibly dominated by vector search and embedding computations, are less sensitive to the textual characteristics of the query within the tested ranges, and that other systemic factors contribute more significantly to the overall latency.
- **Accuracy (is_in_top_5)**:
    - **High Precision for Direct Matches**: For `original`, `short`, and `long` queries, the system consistently returned the expected `unique_id` within the top 5 results, demonstrating excellent recall and precision for well-defined or semantically close inputs.
    - **Correct Handling of Non-Existent Queries**: As expected, `nonexistent` queries correctly yielded no relevant `unique_id` in the top 5, confirming the system's ability to avoid "hallucinations" for out-of-scope requests.
    - **Nuance with Ambiguity**: The mixed results for `ambigious` queries (one successful, one not in top 5) highlight the complexities of semantic search for nuanced inputs. While the system can sometimes resolve ambiguity, its performance in such cases is not always consistent, suggesting that the effectiveness might depend on the specific nuances of the query and the distribution of similar content within the dataset.

Recommendations:

Based on these findings, we propose the following recommendations for optimizing the BookAura Knowledge Base:

1. **Investigate Outlier Causes**: A deeper dive into the root causes of the observed response time outliers is crucial. This could involve detailed logging within MindsDB and

the FastAPI application to identify specific operations (e.g., embedding generation, vector store lookups, network calls) that contribute to these spikes. Monitoring system resources (CPU, RAM, disk I/O) during peak latency events could also provide insights.

2. **Implement Robust Application-Level Handling**: Given the inherent randomness and outliers, consider implementing application-level strategies in the FastAPI backend:
    - **Timeouts**: Ensure appropriate timeouts for MindsDB queries to prevent user-facing latency.
    - **Retries with Backoff**: For critical queries, consider implementing a retry mechanism with exponential backoff to handle transient MindsDB or network issues, improving overall reliability.
    - **Asynchronous Processing**: For very long or complex queries, explore asynchronous processing patterns to prevent the main API thread from blocking.
3. **Optimize MindsDB Configuration**: Review MindsDB's internal configuration parameters related to resource allocation, caching, and concurrent query handling. Experiment with different settings to see if the variability can be reduced or if average performance can be consistently improved.
4. **Explore MindsDB Caching**: If MindsDB offers internal caching mechanisms for frequently accessed queries or specific embedding lookups, investigate enabling and optimizing these to further reduce latency, especially for popular searches.
5. **Qualitative Analysis of Ambiguous Failures**: Conduct a qualitative review of the ambiguous queries that failed to place the expected result in the top 5. This can help understand *why* the semantic match was insufficient and potentially inform strategies for improving the KB's understanding of complex or nuanced inputs, perhaps by refining chunking strategies or exploring different embedding models.
6. **Consider Scalability for Higher Loads**: If future requirements involve significantly higher concurrent query loads or larger datasets, exploring a more distributed MindsDB setup or integrating with dedicated vector databases might be necessary. This stress testing phase will be crucial for understanding the system's true scalability boundaries.

This report provides a foundational understanding of the BookAura search system's performance. Continued monitoring and targeted optimizations based on these insights will be key to enhancing user experience and system reliability.