Prompts_log

📂 AI Interaction Log: MCP CSV Explorer

Project: Analytics Edition

Student: Michele Sagone

ID: 720510

Email: m.sagone1@studenti.unipi.it

🟢 SESSION 1: ARCHITECTURE & SCAFFOLDING

Objective: Establish the server foundation using correct transport protocols.

> **PROMPT 1 (Initialization):**
>
> "Act as a Senior Python Engineer. I need to build a Model Context Protocol (MCP) server that exposes a local directory of CSV files ('./data') to an LLM.
>
> Requirements:
>
> 1. Use the 'fastmcp' SDK.
> 2. Do NOT use stdio transport. Use SSE (Server-Sent Events) with 'uvicorn' because I need to debug this using the MCP Inspector web tool.
> 3. Create a basic 'list_tables' tool to verify the connection."

🔵 SESSION 2: DATA MARSHALLING STRATEGY

Objective: Implement robust data reading.

> **PROMPT 2 (Refining the Stack):**
>
> "Now implement the 'get_schema' and 'query_data' tools.
>
> Constraint: Do not use the built-in 'csv' library because it treats everything as strings. Use 'pandas' instead. I need the server to be aware of data types (int, float) for better LLM reasoning. Return the data in Markdown format."

🟠 SESSION 3: CRITICAL VERIFICATION & OPTIMIZATION

Objective: Fixing inefficient AI-generated logic (The "Human-in-the-Loop" Intervention).

> **PROMPT 3 (Rejection of Iterative Logic):**
>
> "I reviewed your code for 'search_in_table'. You implemented a Python 'for' loop to check every row.
>
> **CRITICAL FEEDBACK:** This is inefficient ($O(n)$ interpreter overhead) and not acceptable for this project.
>
> **CORRECTION:** Rewrite the function using Pandas VECTORIZED operations (e.g., .str.contains). Handle 'NaN' values gracefully so the server doesn't crash on dirty data."

🟣 SESSION 4: ANALYTICS & DETERMINISM

Objective: Offloading math from the LLM to the Server.

> **PROMPT 4 (Adding Deterministic Features):**
>
> "Implement a tool called 'get_stats(table_name)'.
>
> Reasoning: LLMs are bad at arithmetic. I want the server to calculate Mean, Min, and Max deterministically using Pandas and return only the summary to the LLM. This saves tokens and prevents hallucinations."

🔴 SESSION 5: ARCHITECTURAL REFACTORING

Objective: Solving GIL blocking issues and Path Traversal (Concurrency & Safety).

> **PROMPT 5 (The "Advanced Programming" Fix):**
>
> "I noticed a performance issue: Pandas operations are blocking the Async Event Loop, causing the SSE heartbeat to freeze during heavy loads.
>
> 1. Refactor the code to wrap all DataFrame operations in 'asyncio.to_thread' to offload them to a thread pool.

> 2. SECURITY UPDATE: Replace string-based path handling with 'pathlib'. Use '.resolve()' to strictly validate that files are inside the data directory to prevent Path Traversal attacks."

🔘 SESSION 6: CODE HARDENING & TYPE SAFETY

Objective: Meeting professional standards.

> **PROMPT 6 (Final Polish):**
>
> "The code logic is solid, but the style is inconsistent.
>
> 1. Add strict Python Type Hints (e.g., → List[str], → Dict[str, Any]) to ALL functions.
> 2. Add logging instead of print statements.
> 3. Verify that all tools return clear error messages instead of raising unhandled exceptions."

🟡 SESSION 7: METAPROGRAMMING

Objective: Adding Prompt Templates for structured reasoning.

> **PROMPT 7:**
>
> "Add 3 MCP Prompts (@mcp.prompt) to the server:
>
> 1. 'analyze_csv_full': A workflow for a data scientist.
> 2. 'audit_data_quality': To check for missing values.
> 3. 'business_report': To correlate products and orders.
>
>   Ensure the prompts explicitly tell the LLM which tools to call."