Decision Log: Monday.com Business Intelligence Agent

Candidate: Nishanth M
Project: Skylark Drones Technical Assignment

1. Architecture & Tech Stack Selection

- **Framework:** Python + Streamlit.
  - *Decision:* Streamlit was chosen for its native ability to handle dataframes and rapid prototyping. It satisfies the "Hosted Prototype" requirement via Streamlit Cloud and provides a clean, executive-level UI without the overhead of a full-stack JavaScript framework.
- **Intelligence Engine:** Groq (Llama-3.3-70b-versatile).
  - *Decision:* While OpenAI is the industry standard, Groq was selected due to its superior inference speed and the availability of a high-performance free tier. The Llama-3.3-70b model provides the reasoning capabilities necessary to join data across two disparate boards (Work Orders and Deals) effectively.
- **Data Processing:** Pandas.
  - *Decision:* I utilized Pandas to clean and normalize data before it reaches the LLM. This ensures mathematical accuracy (calculating totals/metrics), which LLMs often struggle with if performing raw arithmetic on string data.

2. Monday.com Integration Strategy

- **Method:** Monday.com GraphQL API v2.
  - *Decision:* I implemented a direct API integration using the `requests` library. I avoided third-party MCPs to minimize dependencies and ensure the agent interacts directly with the latest "Items Page" schema.
- **Constraint Handling:** The agent fetches up to 500 items per board. For this prototype, I prioritized fetching the most recent data to stay within LLM context window limits while providing a relevant "snapshot" of the business.

3. Data Resilience & Messy Data Handling

Business data is inherently inconsistent. I addressed this through several layers:

- **Numeric Normalization:** Real-world CSV data often contains symbols (₹), commas (1,000), and placeholder text ("Masked"). I implemented a robust regex-based cleaner that strips non-numeric characters and coerces errors to zero. This prevents app crashes during aggregation.
- **Column Mapping:** Since the Monday.com API returns headers based on user-defined board titles, I used a fuzzy-matching logic (keyword search) to identify "Sector," "Revenue," and "Deal Value" columns dynamically.
- **The "Masked" Data Problem:** I instructed the LLM to acknowledge "Masked" values as data quality caveats. Instead of ignoring them, the agent communicates to the founder that certain insights are based on available (unmasked) figures, ensuring transparency.

4. Interpretation of "Leadership Updates"

I interpreted the requirement for **Leadership Updates** not as a simple list of status updates, but as a **Synthesis of Health**.

- **Executive Focus:** The "Leadership Briefing" feature specifically aggregates **Billed Revenue** (from Work Orders) against **Pipeline Value** (from Deals).
- **Risk Detection:** The agent scans the "Execution Status" and "Nature of Work" columns to identify projects marked as "Stuck" or "Pause/Struck."
- **Actionable Insights:** The update is structured to provide a "Bottom Line Up Front" (BLUF), followed by a Revenue Pulse and a Risk Watch, mirroring the format a Chief of Staff would present to a Founder.

5. Key Assumptions

- **Context Window:** I assumed that providing the top 40-50 rows of data provides enough context for the LLM to identify trends and answer specific queries accurately.
- **Security:** I assumed that API keys should be handled via the frontend sidebar for the prototype, rather than hardcoding, to ensure the app is testable by the recruiter with their own credentials.
- **Currency:** All calculations assume INR (₹) based on the provided sample data headers.

6. Trade-offs & Future Improvements

- **Trade-off (Real-time vs. Sync):** I chose a "Sync Button" approach rather than real-time fetching on every chat message to reduce API latency and avoid rate-limiting on the free tier.
- **What I'd do differently with more time:**
  1. **RAG (Retrieval-Augmented Generation):** For boards with >10,000 items, I would implement a Vector Database (Pinecone) to search through historical records.
  2. **Persistent Chat:** Add SQL-based conversation memory to allow for multi-turn follow-up questions.
  3. **Advanced Visuals:** Integrate a more complex Plotly dashboard with time-series forecasting for projected revenue.
