**🧾 SECTION 1: BASIC DETAILS**

**Name:** Nalla Sai Deepak  
**AI Agent Title / Use Case:** AI Agent for Assisting Data Analysts in Exploring and Summarizing Business Datasets

**🧠 SECTION 2: PROBLEM FRAMING**

**2.1 What problem does your AI Agent solve?**

Data analysts often receive raw datasets in various formats and need to quickly summarize, visualize, or explore them. This agent acts as an assistant that can read multiple file types (CSV, Excel, PDF, DOCX), interpret them, and respond to natural language queries about the data.

**2.2 Why is this agent useful?**

It helps analysts save time by automating exploratory data analysis tasks such as summarizing datasets, identifying trends, suggesting visualizations, and extracting insights — all through a conversational interface.

**2.3 Who is the target user?**

Junior or solo data analysts in a business or startup environment, who need help with quick data reviews and insights without writing full code each time.

**2.4 What not to include?**

* Advanced statistical modeling (e.g., regressions, clustering)
* Database connections or SQL querying
* Deep learning or forecasting

**2.5 What dataset(s) did you use?**

* **metadata.csv**: A dataset of tech product specifications, including fields such as battery, memory, processor, camera, and price.

**🧱 SECTION 3: 4-LAYER PROMPT DESIGN**

**🔹 3.1 INPUT UNDERSTANDING**

**Prompt:**  
"User may upload a file or ask a question. If a file is uploaded, identify its type. If a text query is present, interpret the intent — is the user asking for a summary, a chart, or insight?"

**What is this prompt responsible for?**  
Detecting user intent and type of request.

**Example Input + Output:**  
Input: *"Here's a CSV file, can you summarize it?"*  
Output: *"Sure, summarizing uploaded CSV file..."*

**🔹 3.2 STATE TRACKER**

**Prompt:**  
"If a file is uploaded, store metadata like file name, type, and data preview. Track user’s last question to maintain continuity."

**How does this help the agent “remember”?**  
Uses Python variables to store state (uploaded\_file, last\_query). System messages help ChatGPT recall user context in ongoing sessions.

**Did you simulate memory?**  
Yes, through variable tracking and prompt reinforcement in system messages.

**🔹 3.3 TASK PLANNER**

**Prompt:**  
"If file type is CSV/Excel, load it into pandas and identify key columns. If the user asks for a summary, generate value counts, mean, std, etc. If charts are requested, suggest the most relevant visualizations."

**What steps does your agent take?**

* Parse file type
* Load using relevant library
* Choose analysis task (summary vs chart)
* Generate Python code behind the scenes
* Return result in human-readable format

**Did you use chaining or branching?**  
Yes — file parsing branches into multiple task paths (summarization, visualization, text extraction).

**🔹 3.4 OUTPUT GENERATOR**

**Prompt:**  
"Convert analysis or plot suggestions into simple English. Use markdown for structure. Output charts with alt-text. For text files, summarize key bullet points."

**Formatting / special behavior?**

* Used markdown for clarity
* Fallback messages for empty files or errors
* Context-aware tone ("Here’s what I found in your file...")

**🔍 SECTION 4: CHATGPT EXPLORATION LOG**

| **Attempt** | **Prompt Variant** | **What Happened** | **What You Changed** | **Why You Changed It** |
| --- | --- | --- | --- | --- |
| 1 | “Summarize my file” | Output too vague | Added system prompt to detect file type | Clarified task intent |
| 2 | “What chart can I make from this?” | Gave generic answer | Added column inspection logic | To suggest context-relevant plots |
| 3 | “Here’s a PDF, analyze it” | Only returned text | Added PyMuPDF + summary steps | To extract usable content |
| 4 | “Give me insights from this Excel” | Gave summary | Good | — |

**🧪 SECTION 5: OUTPUT TESTS**

**Test 1: Normal input (CSV)**

*Dataset: metadata.csv*  
**Input:** “Here’s a dataset, summarize it.”  
**Output:**

* File has 6 columns and 20 rows.
* Columns include Battery, Memory, Processor, Price.
* Mean price: 24500.
* Suggested plots: Battery vs Price, Memory vs Price.

**Test 2: Vague Input (no file)**

**Input:** “Tell me something useful.”  
**Output:**

* “Please upload a dataset or ask a specific question about your data.”

**Test 3: Invalid input (image file)**

**Input:** Uploaded a .png  
**Output:**

* “I can’t extract data from image files yet. Please upload a CSV, Excel, PDF, or DOCX file.”

**🔄 SECTION 6: REFLECTION**

**6.1 What was the hardest part?**  
Figuring out how to generalize prompts for different file types. Also, simulating memory and flow in ChatGPT was challenging.

**6.2 What did you enjoy most?**  
Building an AI that feels like a helpful teammate. I enjoyed testing with real files and seeing meaningful summaries come back.

**6.3 What would you improve?**  
Add file validation, better error handling, and a GUI for uploading files instead of relying on manual input.

**6.4 What did you learn about ChatGPT or prompt design?**  
That breaking prompts into specific roles helps get better results. Also, iteration and test-driven debugging is essential.

**6.5 Did you ever feel stuck?**  
Yes, especially when file types were not parsed correctly. I debugged by adding clear conditionals and testing with known files.

**🧠 SECTION 7: HACK VALUE (Optional)**

* Added support for multiple file types: CSV, Excel, PDF, DOCX
* Simulated memory via variables
* Used pandas, PyMuPDF, and Python logic to simulate backend
* Built a robust backend agent structure with minimal user effort