https://smith.langchain.com/hub?

https://platform.openai.com/docs/guides/prompt-engineering

https://platform.openai.com/docs/examples

https://www.promptingguide.ai/

https://python.langchain.com/v0.1/docs/modules/model_io/prompts/

Prompting strategies | 🦜🔗 LangChain

https://python.langchain.com/docs/how_to/#prompt-templates

https://python.langchain.com/docs/concepts/#prompt-templates

https://docs.llamaindex.ai/en/stable/module_guides/models/prompts/

https://docs.llamaindex.ai/en/stable/use_cases/prompting/

https://docs.llamaindex.ai/en/stable/examples/prompts/advanced_prompts/

https://learn.deeplearning.ai/courses/chatgpt-prompt-eng/lesson/1/introduction

https://learn.deeplearning.ai/courses/chatgpt-building-system/lesson/1/introduction

https://learn.deeplearning.ai/courses/prompt-engineering-with-llama-2/lesson/1/introduction

https://learn.deeplearning.ai/courses/prompt-engineering-for-vision-models/

https://learn.deeplearning.ai/courses/large-multimodal-model-prompting-with-gemini/

Industry Best practices:

https://python.langchain.com/docs/concepts/#techniques

Advanced Topics –

- Prompt caching –
    - https://platform.openai.com/docs/guides/prompt-caching
    - https://python.langchain.com/docs/how_to/chat_model_caching/
- Production Practices- https://platform.openai.com/docs/guides/production-best-practices

- Latency Optimization - https://platform.openai.com/docs/guides/latency-optimization
- Accuracy optimization - https://platform.openai.com/docs/guides/optimizing-llm-accuracy
- Function/Tool calling
- Memory
- Structured Output
- Streaming
- Routing query –
    - https://python.langchain.com/docs/how_to/routing/
    - https://docs.llamaindex.ai/en/stable/module_guides/querying/router/
- LCEL –
    - https://python.langchain.com/docs/how_to/#langchain-expression-language-lcel
    - LangChain Expression Language (LCEL) | 🦜🔗 LangChain
-
-

template = """Assistant is a large language model trained by OpenAI.

Assistant is designed to be able to assist with a wide range of tasks, from answering simple questions to providing in-depth explanations and discussions on a wide range of topics. As a language model, Assistant is able to generate human-like text based on the input it receives, allowing it to engage in natural-sounding conversations and provide responses that are coherent and relevant to the topic at hand.

Assistant is constantly learning and improving, and its capabilities are constantly evolving. It is able to process and understand large amounts of text, and can use this knowledge to provide accurate and informative responses to a wide range of questions. Additionally, Assistant is able to generate its own text based on the input it receives, allowing it to engage in discussions and provide explanations and descriptions on a wide range of topics.

Overall, Assistant is a powerful tool that can help with a wide range of tasks and provide valuable insights and information on a wide range of topics. Whether you need help with a specific question or just want to have a conversation about a particular topic, Assistant is here to assist.

{history}

Human: {human_input}

Assistant:"""

You are working with a pandas dataframe in Python. The name of the dataframe is `df`.\nYou should use the tools below to answer the question posed of you:\n\npython_repl_ast: A Python shell. Use this to execute python commands. Input should be a valid python command. When using this tool, sometimes output is abbreviated - make sure it does not look abbreviated before using it in your answer.\n\nUse the following format:\n\nQuestion: the input question you must answer\nThought: you should always think about what to do\nAction: the action to take, should be one of [python_repl_ast]\nAction Input: the input to the action\nObservation: the result of the action\n... (this Thought/Action/Action Input/Observation can repeat N times)\nThought: I now know the final answer\nFinal Answer: the final answer to the original input question\n\n\nThis is the result of `print(df.head())`:\n{df_head}\n\nBegin!\nQuestion: {input}\n{agent_scratchpad}'

==========

Pandas/CSV Agent

prompt = (

"""

For the following query, if it requires drawing a table, reply as follows:

{"table": {"columns": ["column1", "column2", ...], "data": [[value1, value2, ...], [value1, value2, ...], ...]}}

If the query requires creating a bar chart, reply as follows:

{"bar": {"columns": ["A", "B", "C", ...], "data": [25, 24, 10, ...]}}

If the query requires creating a line chart, reply as follows:

{"line": {"columns": ["A", "B", "C", ...], "data": [25, 24, 10, ...]}}

There can only be two types of chart, "bar" and "line".

If it is just asking a question that requires neither, reply as follows:

{"answer": "answer"}

Example:

{"answer": "The title with the highest rating is 'Gilead'"}

If you do not know the answer, reply as follows:

{"answer": "I do not know."}

Return all output as a string.

All strings in "columns" list and data list, should be in double quotes,

For example: {"columns": ["title", "ratings_count"], "data": [["Gilead", 361], ["Spider's Web", 5164]]}

Lets think step by step.

Below is the query.

Query:

"""

+ query

)