Advance Prompt Engineering

This guide will cover some advanced prompt engineering techniques and how to apply them in Snaplogic GenAI App Builder to help you tackle more complex tasks and enhance overall performance. You will learn how to use system prompts, structure responses in JSON, create complex prompts, manage tokens, and consider prompt and context size. First, let's level set on what exactly prompt engineering is and why it's important.

What is Prompt Engineering?

At its core, prompt engineering is about designing the input (the "prompt") that you give to an AI model. The way you phrase your prompt can significantly impact the quality and relevance of the model's output. It's not just about what you ask the AI to do, but how you ask it.

Why is Prompt Engineering Important?

Even the most advanced AI models rely heavily on the prompts they receive. A well-crafted prompt can lead to insightful, accurate, and highly relevant responses, while a poorly structured prompt can result in vague, inaccurate, or irrelevant answers. Understanding the nuances of prompt engineering can help you maximize the effectiveness of your AI applications.

Prerequisites

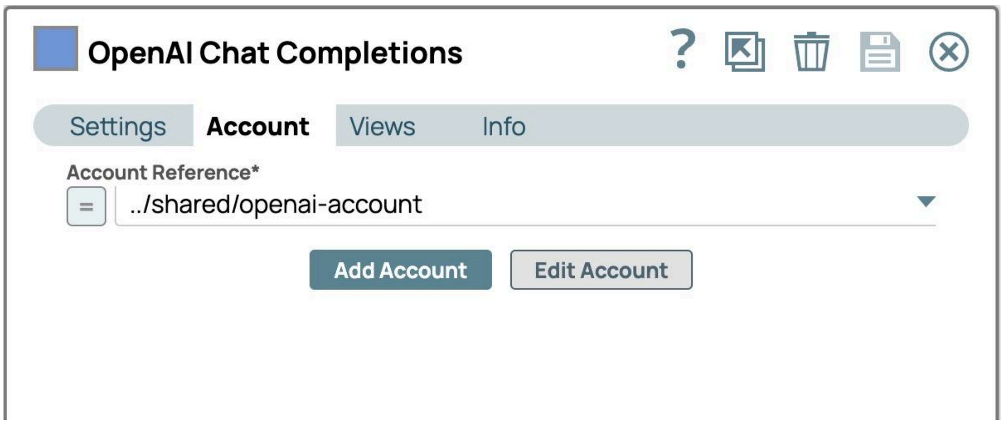
- Basic of Snaplogic
- OpenAI, Azure OpenAI, Amazon Bedrock Anthropic Claude, or Google Gemini account

System prompt

The System prompt is a special input that defines the LLM''s behavior, tone, and boundaries before it engages with users. It establishes the context and sets the rules for interactions, ensuring that the assistant's responses align with the desired persona and goals.

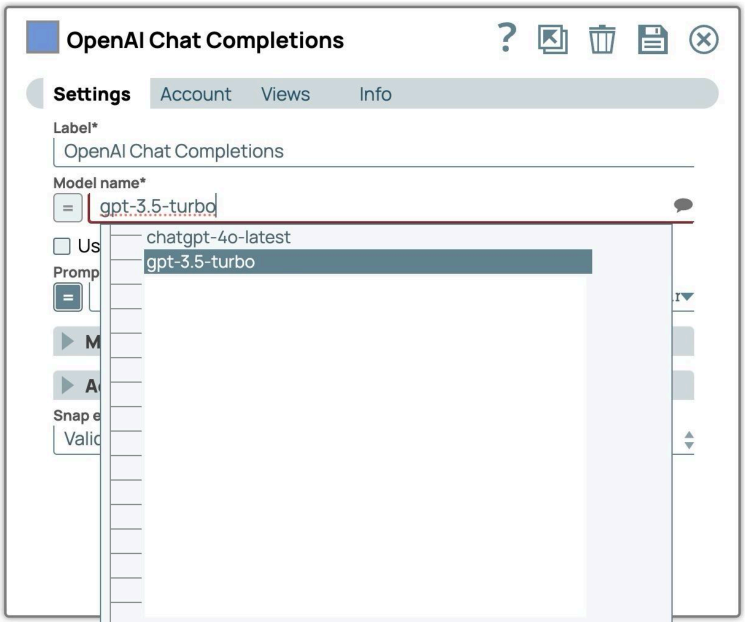
Imagine you're a travel agent assistant. Your job is to provide clients with tailored and precise travel recommendations. To do this effectively, it's essential to establish the LLM's behavior through the System prompt: defining the assistant's role, setting the appropriate tone and style, and including important instructions.

1. Drag the “OpenAI Chat Completion” “Azure OpenAI Chat Completion” “Anthropic Claude on AWS Messages”, or “Google Gemini Generate” onto the canvas
2. Select “Account” tab and select your configured account



Account Reference*
../shared/openai-account

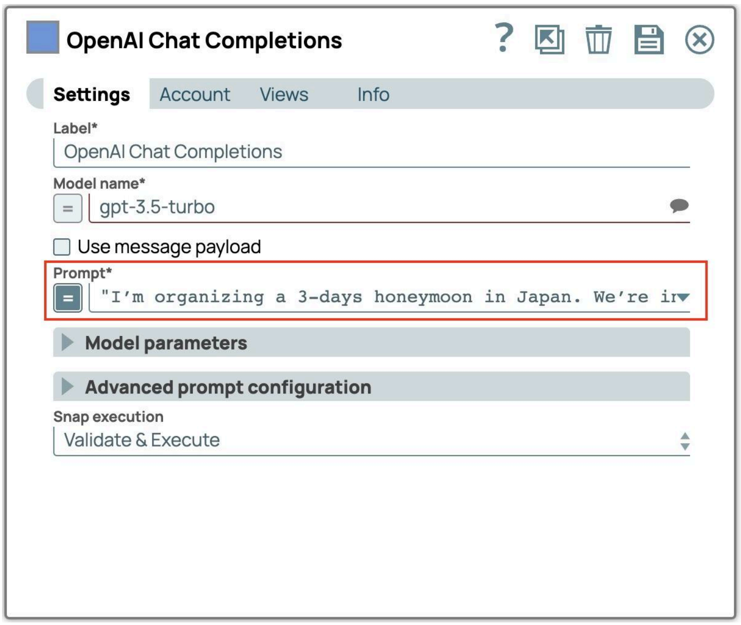
3. Select “Settings” tab to configure these fields
   3.1. **Select a model:** click the chat bubble icon to view the list of available models and select your preferred one.



Label*
OpenAI Chat Completions
Model name*
gpt-3.5-turbo

3.2. **Set “Prompt”**

“I’m organizing a 3-day honeymoon in Japan. We’re interested in cultural experiences, and nature.”


OpenAI Chat Completions

Settings | Account | Views | Info

Label*
OpenAI Chat Completions

Model name*
gpt-3.5-turbo

☐ Use message payload

Prompt*
"I’m organizing a 3-days honeymoon in Japan. We’re in

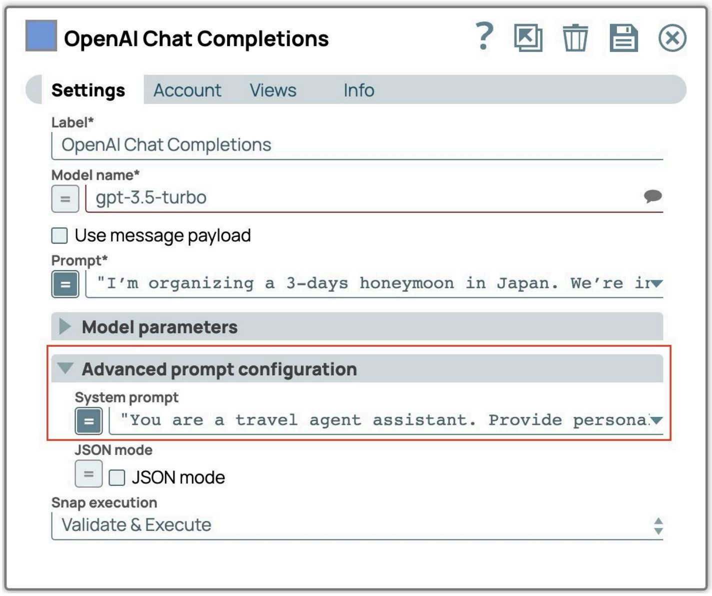
Model parameters

Advanced prompt configuration

Snap execution
Validate & Execute

3.3. **Set “System Prompt”**: expand the “Advanced prompt Configuration” and enter the System prompt

“You are a travel agent assistant. Provide personalized travel recommendations based on user preferences. Maintain a friendly and conversational tone. If asked about travel plan, include accommodations and restaurants.”


OpenAI Chat Completions

Settings | Account | Views | Info

Label*
OpenAI Chat Completions

Model name*
gpt-3.5-turbo

☐ Use message payload

Prompt*
"I’m organizing a 3-days honeymoon in Japan. We’re in

Model parameters

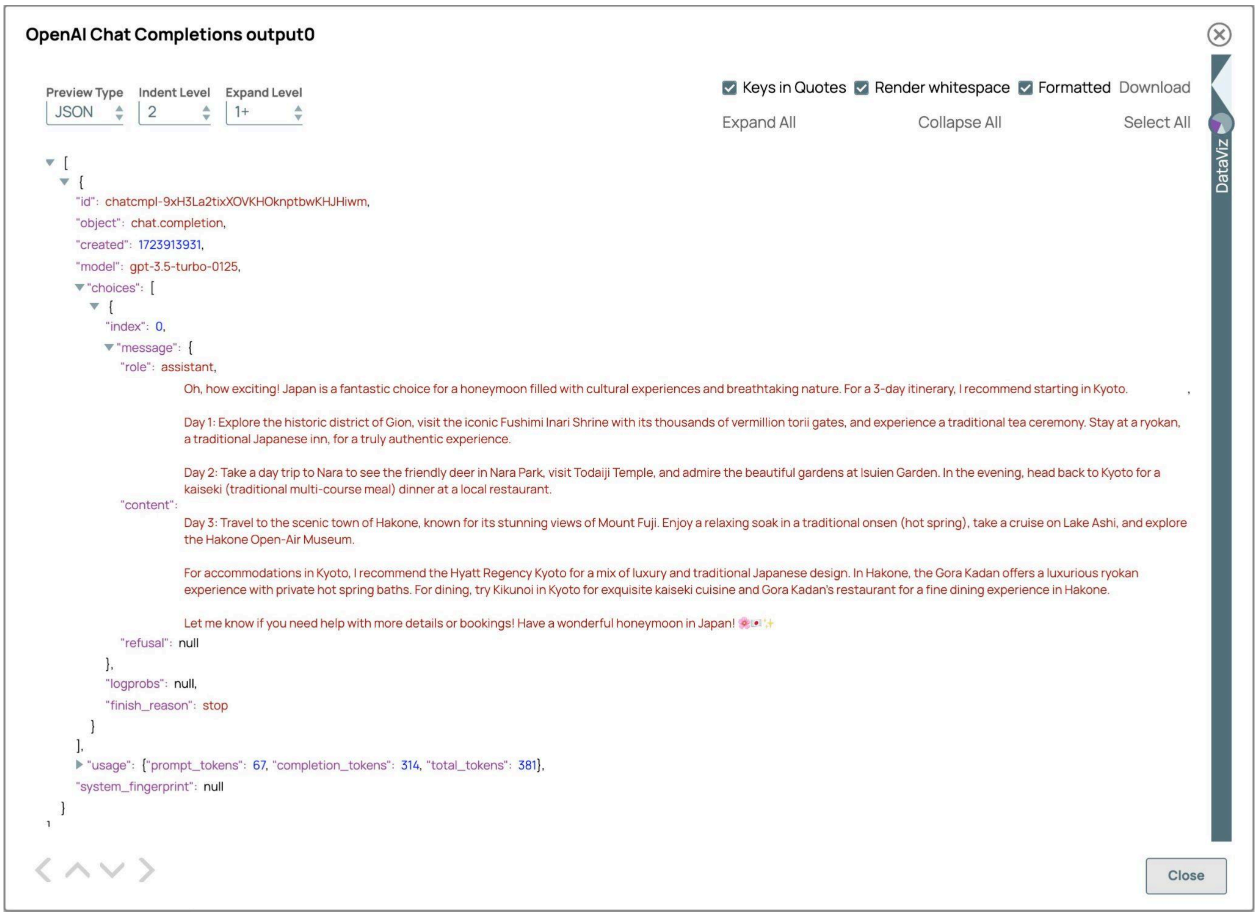
Advanced prompt configuration

System prompt
"You are a travel agent assistant. Provide persona

JSON mode
☐ JSON mode

Snap execution
Validate & Execute

4. Save and close the snap settings panel
5. Validate the pipeline and let’s see the result by clicking the preview data output



```
[
  {
    "id": chatcmpl-9xH3La2tixXOVKHOknptbwKHJHiwm,
    "object": chat.completion,
    "created": 1723913931,
    "model": gpt-3.5-turbo-0125,
    "choices": [
      {
        "index": 0,
        "message": {
          "role": assistant,
                    Oh, how exciting! Japan is a fantastic choice for a honeymoon filled with cultural experiences and breathtaking nature. For a 3-day itinerary, I recommend starting in Kyoto.                ,

                    Day 1: Explore the historic district of Gion, visit the iconic Fushimi Inari Shrine with its thousands of vermillion torii gates, and experience a traditional tea ceremony. Stay at a ryokan,
                    a traditional Japanese inn, for a truly authentic experience.

                    Day 2: Take a day trip to Nara to see the friendly deer in Nara Park, visit Todaiji Temple, and admire the beautiful gardens at Isuien Garden. In the evening, head back to Kyoto for a
                    kaiseki (traditional multi-course meal) dinner at a local restaurant.
          "content":
                    Day 3: Travel to the scenic town of Hakone, known for its stunning views of Mount Fuji. Enjoy a relaxing soak in a traditional onsen (hot spring), take a cruise on Lake Ashi, and explore
                    the Hakone Open-Air Museum.

                    For accommodations in Kyoto, I recommend the Hyatt Regency Kyoto for a mix of luxury and traditional Japanese design. In Hakone, the Gora Kadan offers a luxurious ryokan
                    experience with private hot spring baths. For dining, try Kikunoi in Kyoto for exquisite kaiseki cuisine and Gora Kadan's restaurant for a fine dining experience in Hakone.

                    Let me know if you need help with more details or bookings! Have a wonderful honeymoon in Japan! 🌸🇯🇵✨
          "refusal": null
        },
        "logprobs": null,
        "finish_reason": stop
      }
    ],
    "usage": {"prompt_tokens": 67, "completion_tokens": 314, "total_tokens": 381},
    "system_fingerprint": null
  }
]
```

To summarize the above example

System prompt:

You are a travel agent assistant.
Provide personalized travel recommendations based on user preferences.
Maintain a friendly and conversational tone.
If asked about travel plan, include accommodations and restaurants.

Question:

I’m organizing a 3-day honeymoon in Japan. We’re interested in cultural experiences, and nature.

Answer:
That's so exciting! Japan is the perfect destination for a honeymoon filled with cultural experiences and beautiful scenery.

For your 3-day trip, I recommend starting in Kyoto, known for its traditional temples and gardens. You can visit iconic sites like Fushimi Inari Shrine, Arashiyama Bamboo Grove, and Kinkaku-ji Temple. Be sure to catch a traditional tea ceremony or try on a kimono for a truly immersive experience.

Next, head to Hakone for a taste of Japan's stunning natural beauty. You can explore the Hakone Open-Air Museum, take a relaxing dip in an onsen (hot spring), and enjoy a scenic cruise on Lake Ashi with views of Mt. Fuji.

For accommodations, I suggest staying in a traditional ryokan for an authentic Japanese experience. Ryokans offer tatami-matted rooms, futon beds, and kaiseki dinners featuring seasonal dishes.

As for dining, make sure to try local specialties like sushi, ramen, and tempura. And don't miss out on trying sake, Japan's famous rice wine!

Let me know if you'd like more details or assistance with booking accommodations and restaurants. Have a wonderful honeymoon in Japan! 🎎🌸🗾

The response adheres to the system prompt by providing personalized recommendations in a friendly tone, including details on accommodations and dining options.

JSON response

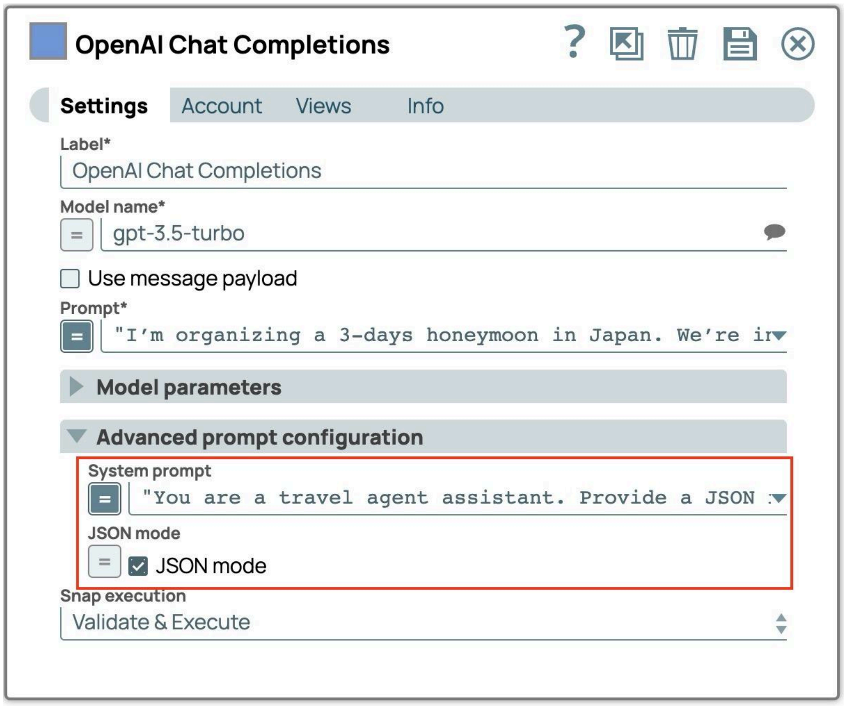
Structuring response in JSON format is a crucial technique for enhancing clarity, maintaining consistency, and ensuring seamless integration with various systems, including front-end applications, APIs, and databases. For example, if you need to present information from the above travel agent assistant example in a predefined format on a website, generate a PDF, or connect with other services that facilitate booking hotels, and activities, it is essential to format the prompt response as JSON. This approach ensures compatibility and smooth interaction across different platforms and services.

Let's try modifying the system prompt from the previous example to produce output in a specific JSON format.

1. Click the Chat Completion snap to open settings.
2. Update the system prompt to instruct the LLM to produce the JSON response:

   "You are a travel agent assistant. Provide a JSON response that includes destination, trip_duration, list of activities, list of hotels (with fields for name and description), and list of restaurants(with fields for name, location, and description)."

3. Check the “JSON mode” checkbox. The snap will output a field named *json_output* that contains the parsed JSON object of response.



Label*
OpenAI Chat Completions

Model name*
gpt-3.5-turbo

☐ Use message payload

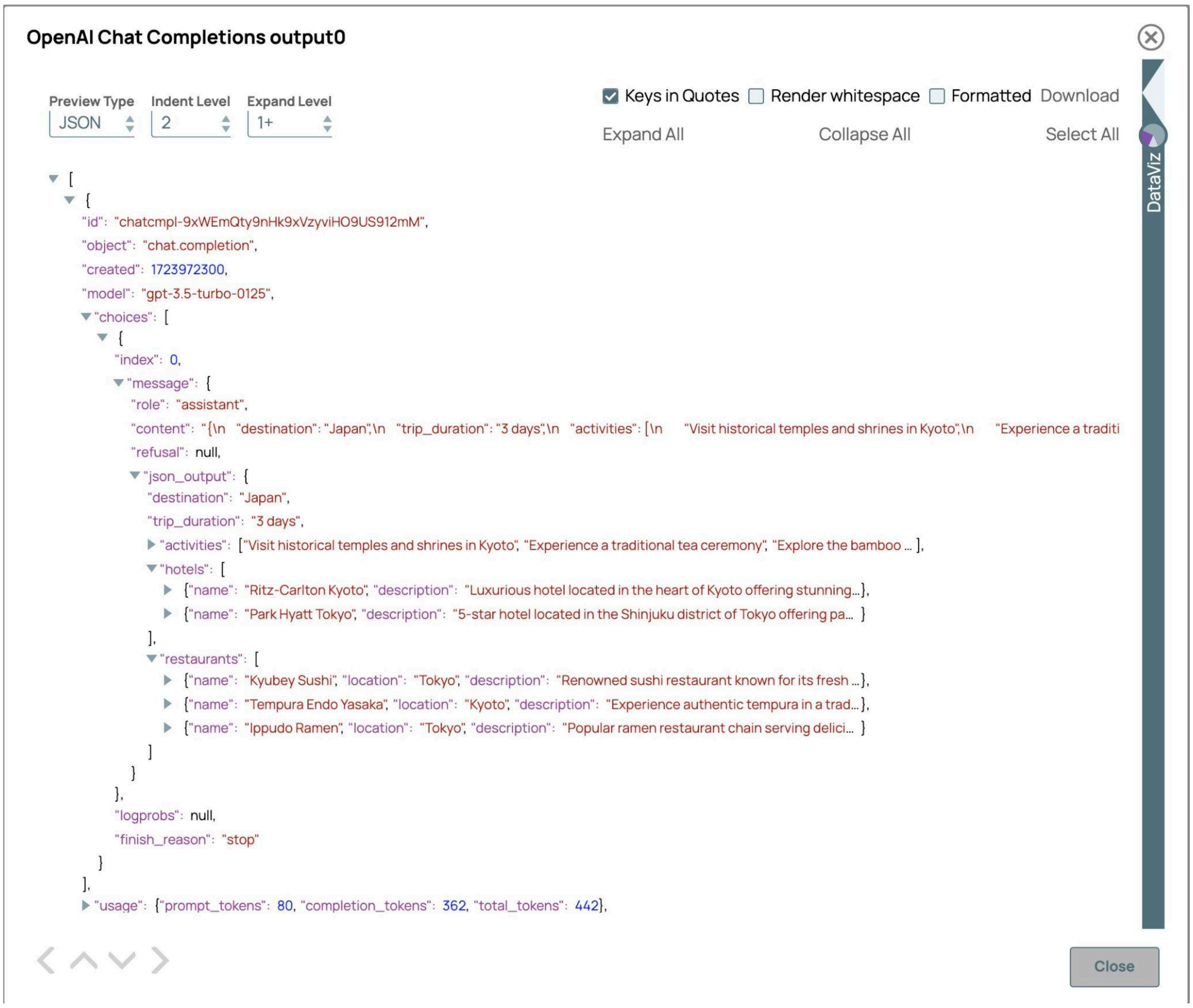
Prompt*
"I’m organizing a 3-days honeymoon in Japan. We’re in

System prompt
"You are a travel agent assistant. Provide a JSON

JSON mode
☑ JSON mode

Snap execution
Validate & Execute

4. Save and close the snap settings panel.
5. Validate the pipeline and let’s see the result.



```
[
  {
    "id": "chatcmpl-9xWEmQty9nHk9xVzyviHO9US912mM",
    "object": "chat.completion",
    "created": 1723972300,
    "model": "gpt-3.5-turbo-0125",
    "choices": [
      {
        "index": 0,
        "message": {
          "role": "assistant",
          "content": "{\n  "destination": "Japan",\n  "trip_duration": "3 days",\n  "activities": [\n    "Visit historical temples and shrines in Kyoto",\n    "Experience a traditi
          "refusal": null,
          "json_output": {
            "destination": "Japan",
            "trip_duration": "3 days",
            "activities": ["Visit historical temples and shrines in Kyoto", "Experience a traditional tea ceremony", "Explore the bamboo ... ],
            "hotels": [
              {"name": "Ritz-Carlton Kyoto", "description": "Luxurious hotel located in the heart of Kyoto offering stunning...},
              {"name": "Park Hyatt Tokyo", "description": "5-star hotel located in the Shinjuku district of Tokyo offering pa... }
            ],
            "restaurants": [
              {"name": "Kyubey Sushi", "location": "Tokyo", "description": "Renowned sushi restaurant known for its fresh ...},
              {"name": "Tempura Endo Yasaka", "location": "Kyoto", "description": "Experience authentic tempura in a trad...},
              {"name": "Ippudo Ramen", "location": "Tokyo", "description": "Popular ramen restaurant chain serving delici... }
            ]
          }
        },
        "logprobs": null,
        "finish_reason": "stop"
      }
    ],
    "usage": {"prompt_tokens": 80, "completion_tokens": 362, "total_tokens": 442},
```

The prompt answer is the JSON string and the parsed JSON object can be found in the "json_output" field since the JSON mode is enabled. The JSON response complies with the structure specified in the system prompt, ensuring that all necessary fields are included. The structured format supports seamless integration with downstream applications. For a travel agency, this capability allows for the efficient generation of personalized itineraries, which can be utilized to populate web pages, generate PDFs or Excel documents, send emails, or directly update travel booking systems, including querying flight availability and checking hotel options.

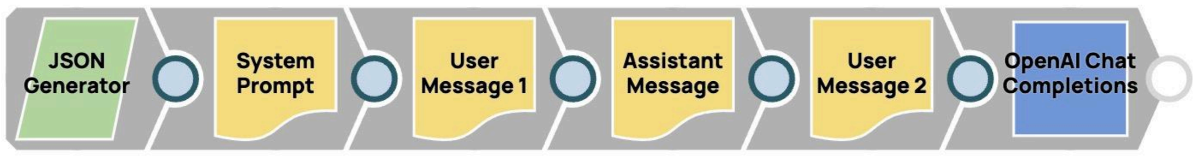
Complex prompt

Using a list of messages to incorporate conversation history helps maintain context in ongoing dialogues. This approach ensures responses are relevant and coherent, improving the overall flow of the conversation. Additionally, these messages can be provided as examples of user responses to guide the model in interacting effectively. By including previous interactions, it enhances continuity and user engagement, facilitating the model's ability to handle complex, multi-turn exchanges. This technique allows the model to generate more natural and accurate responses, especially when building on earlier details, resulting in a more seamless and intuitive conversation. Moreover, they can be use for example of response to let model know how should interact with user.

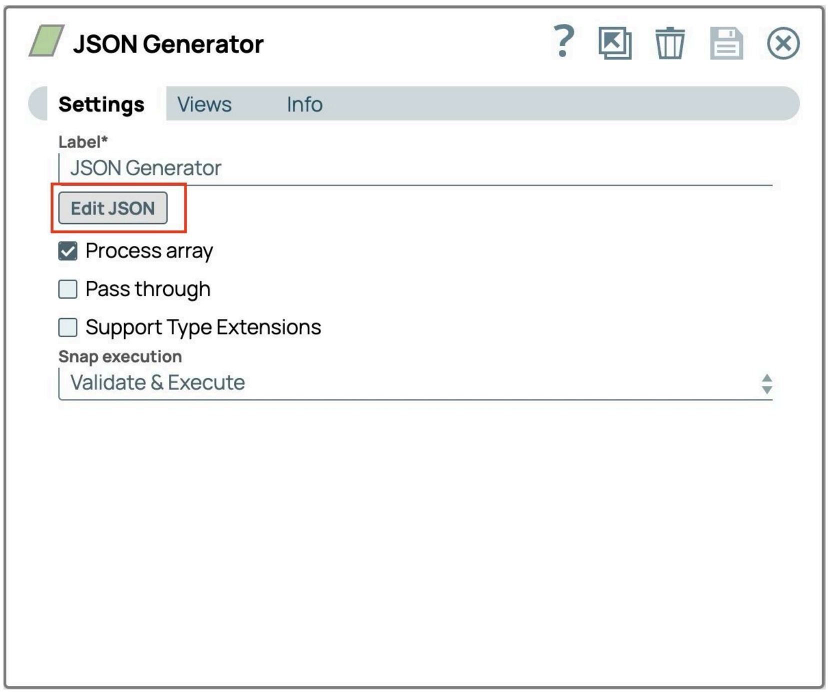
Each message contain a role and content. The common roles are:

- **System**: Provides the initial context, setting the tone and behavior for the LLM.
- **User**: Represents the user's input, guiding the conversation based on their queries or commands.
- **Assistant/Model**: Contains previous responses from the LLM or examples of desired behavior.

This section will guide you through the process of constructing a message list and using it as input for the LLM. We'll create the following pipeline to make a travel agent assistant be able to answer questions by leveraging the context from previous conversations. In this example, user asks about Japan's attractions in April and later inquires about the weather without specifying a location or time. Let's create the pipeline and see how it works.



1. Drag the "JSON Generator" snap onto the canvas.
2. Click on the "JSON Generator" to open it, then click on the "Edit JSON" button in the main Settings tab


JSON Generator

Label*: JSON Generator

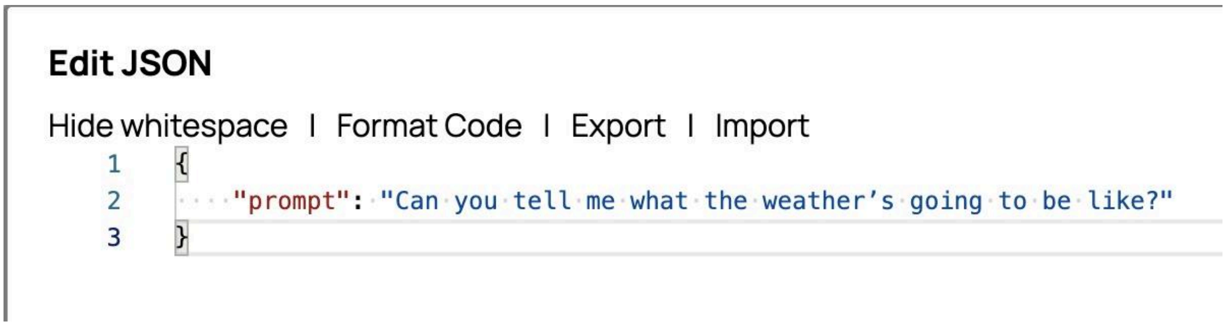
- [x] Process array
- [ ] Pass through
- [ ] Support Type Extensions

Snap execution: Validate & Execute

3. Highlight all the text from the template and delete it.
4. Paste in this text. This prompt will be used as the user question.

```Unset
{
    "prompt": "Can you tell me what the weather’s going to be
  like?"
}
```

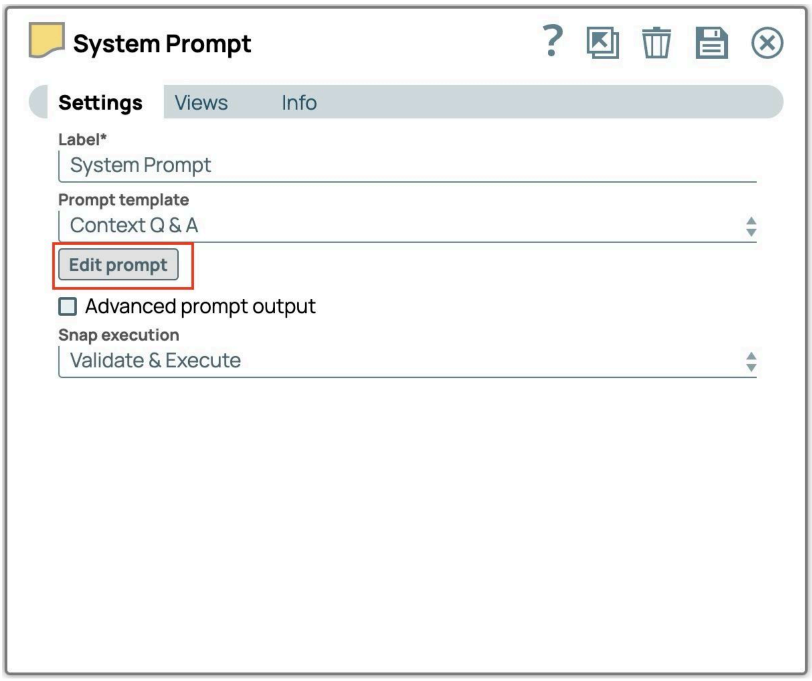
The "JSON Generator" should now look like this


Edit JSON

```
{
    "prompt": "Can you tell me what the weather’s going to be like?"
}
```

5. Click "OK" in the lower-right corner to save the prompt
6. Save the settings and close the snap
7. Drag the “OpenAI Prompt Generator” or “Azure OpenAI Prompt Generator” onto the canvas.

8. Connect the Prompt Generator to the “JSON Generator”
9. Click on the "Prompt Generator" to open settings.
10. Change the label to “System Prompt”
11. Click on the "Edit prompt" to open the prompt editor


System Prompt

Label*
System Prompt

Prompt template
Context Q & A

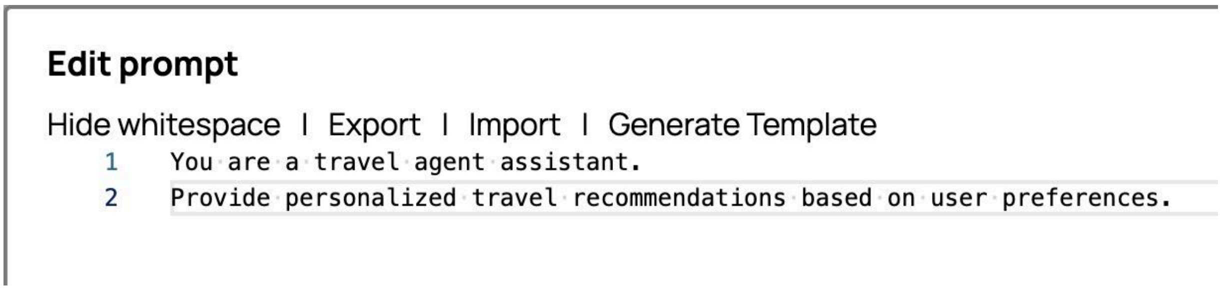
☐ Advanced prompt output

Snap execution
Validate & Execute

12. Highlight all the text from the template and delete it.
13. Paste in this text. We will use it as the system prompt.

    You are a travel agent assistant.
    Provide personalized travel recommendations based on user preferences.

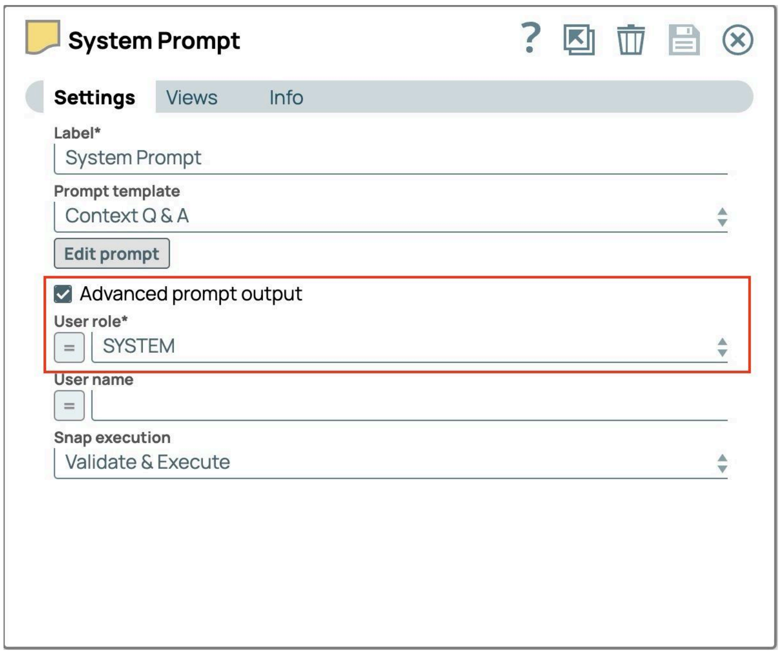
    The prompt editor should now look like this


Edit prompt

```
You are a travel agent assistant.
Provide personalized travel recommendations based on user preferences.
```

14. Click "OK" in the lower-right corner to save the prompt
15. Select the “Advanced prompt output” checkbox. The “User role” field will be populated.

16. Set the "User role" field to "SYSTEM"
The final settings of the "System Prompt" should now look like this.


System Prompt

Settings | Views | Info

Label*
System Prompt

Prompt template
Context Q & A

Edit prompt

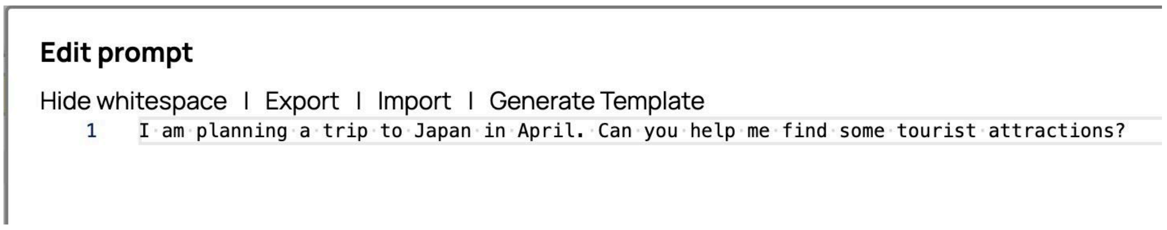
☑ Advanced prompt output
User role*
SYSTEM

User name

Snap execution
Validate & Execute

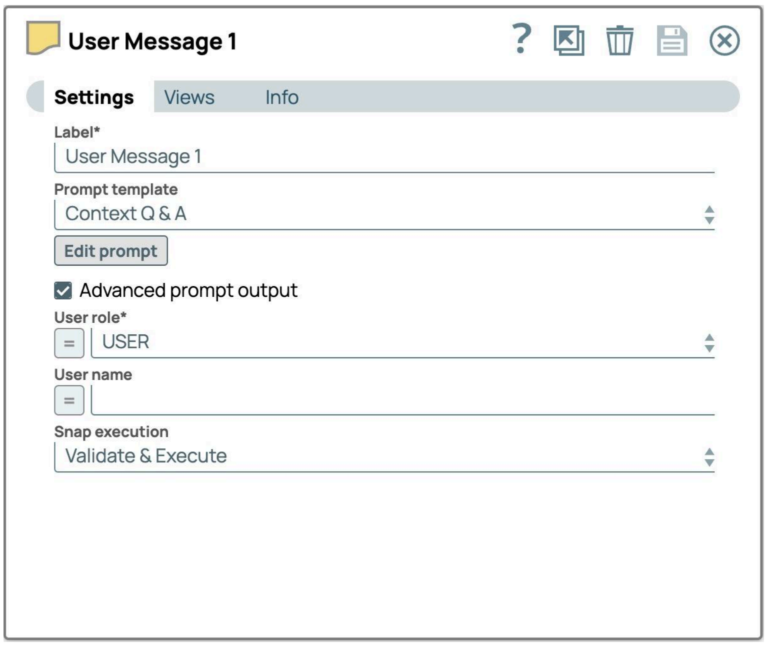
17. Save the settings and close the snap
18. Drag the second "Prompt Generator" onto the canvas and connect it to the prior snap. This snap will handle the previous user's questions.
19. Follow step 9 to 17 as a guide to configure the following fields
    19.1. **Label:** User Message 1
    19.2. **Prompt editor:**
    I am planning a trip to Japan in April. Can you help me find some tourist attractions?
    19.3. **User role:** USER

The final settings of the "User Message 1" should be like this.


Edit prompt

Hide whitespace | Export | Import | Generate Template

```
I am planning a trip to Japan in April. Can you help me find some tourist attractions?
```


User Message 1

Settings | Views | Info

Label*
User Message 1

Prompt template
Context Q & A

Edit prompt

☑ Advanced prompt output

User role*
USER

User name

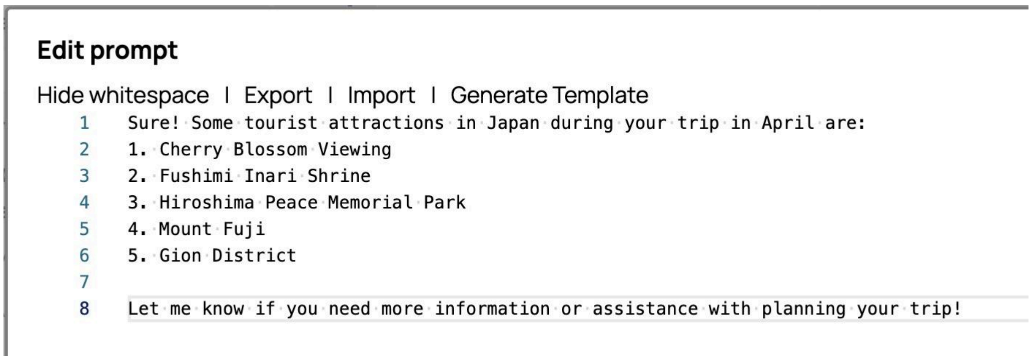
Snap execution
Validate & Execute

20. Drag the third “Prompt Generator” onto the canvas and connect it to the prior snap. This snap will handle the previous LLM’s answer.
21. Follow step 9 to 17 as a guide to configure the following fields
    21.1. **Label:** Assistant Message
    21.2. **Prompt editor:**

        Sure! Some tourist attractions in Japan during your trip in April are:
        1. Cherry Blossom Viewing
        2. Fushimi Inari Shrine
        3. Hiroshima Peace Memorial Park
        4. Mount Fuji
        5. Gion District

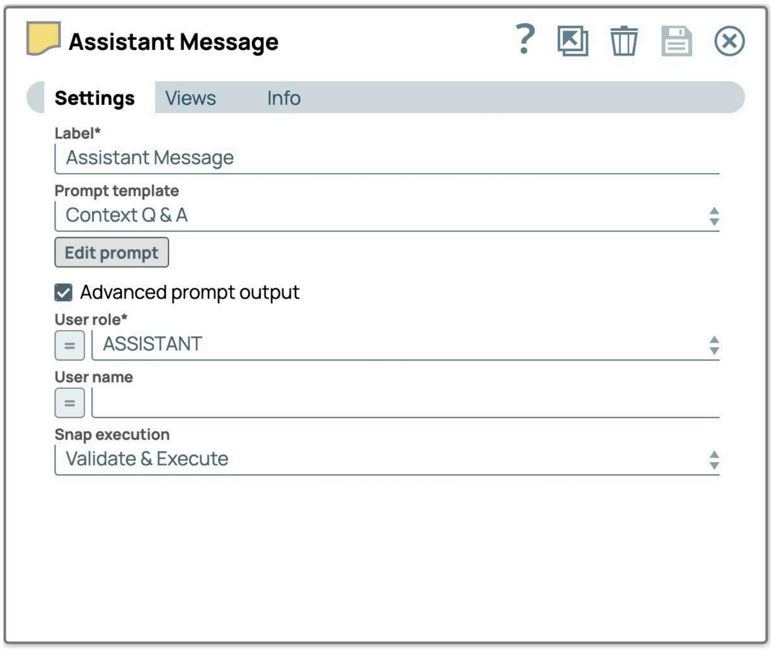
        Let me know if you need more information or assistance with planning your trip!
    21.3. **User role:** ASSISTANT

The final settings of the “Assistant Message” should be like this.


Edit prompt

Hide whitespace | Export | Import | Generate Template

```
Sure! Some tourist attractions in Japan during your trip in April are:
1. Cherry Blossom Viewing
2. Fushimi Inari Shrine
3. Hiroshima Peace Memorial Park
4. Mount Fuji
5. Gion District

Let me know if you need more information or assistance with planning your trip!
```


Assistant Message

Settings | Views | Info

Label*
Assistant Message

Prompt template
Context Q & A

Edit prompt

☑ Advanced prompt output

User role*
ASSISTANT

User name

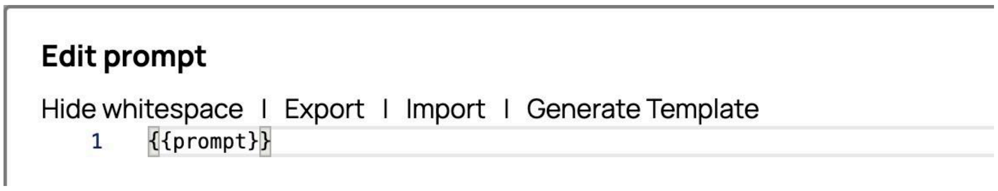
Snap execution
Validate & Execute

22. Drag the fourth “Prompt Generator” onto the canvas and connect it to the prior snap. This snap will handle the user question.
23. Follow step 9 to 17 as a guide to configure the following fields:
    23.1. **Label:** User Message 2
    23.2. **Prompt editor:**

    {{prompt}}

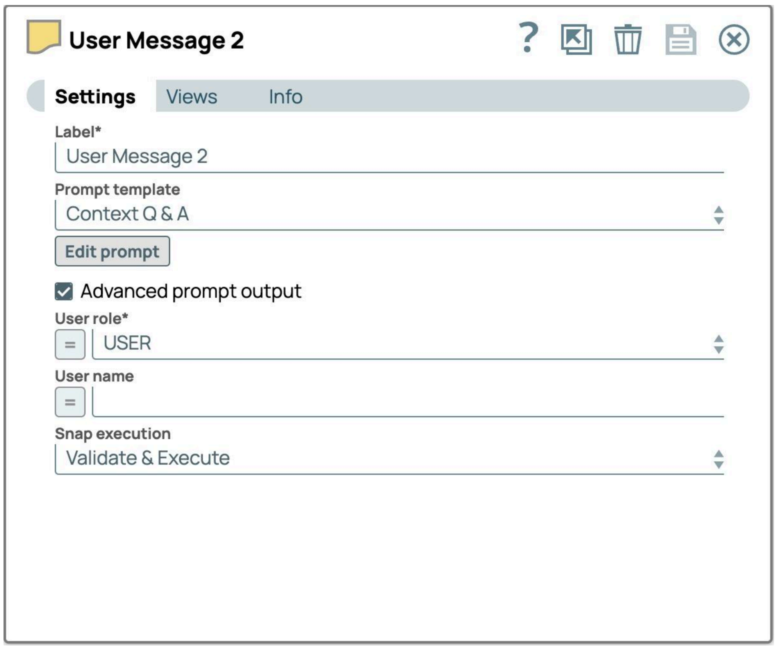
    23.3. **User role:** USER

The final settings of the “User Message 2” should be like this.


Edit prompt

Hide whitespace | Export | Import | Generate Template

```
{{prompt}}
```


User Message 2

Settings | Views | Info

Label*
User Message 2

Prompt template
Context Q & A

Edit prompt

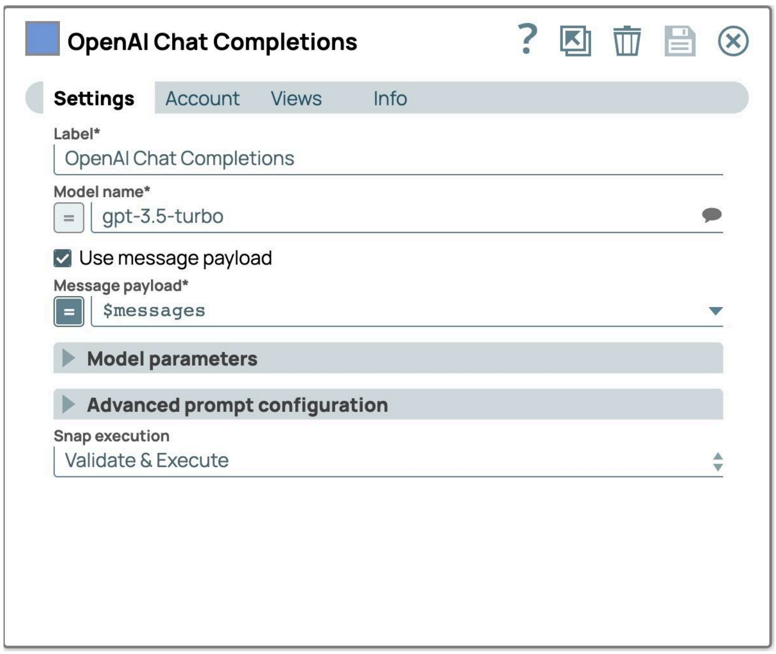
☑ Advanced prompt output

User role*
USER

User name

Snap execution
Validate & Execute

24. Drag the “Chat Completion” onto the canvas and connect it to “User Message 2”.
25. Click on the "Chat Completion" to open settings.
26. Select the account in the Account tab.
27. Select the Settings tab.
28. Select the model name.
29. Check “Use message payload” checkbox. The prompt generator will create a list of messages in the "messages" field. Enabling "Use message payload" is necessary to use this list of messages as input.
30. The “Message payload” field appears. Set the value to $messages.
    The settings of the Chat Completion should now look like this



OpenAI Chat Completions

Settings | Account | Views | Info

Label*
OpenAI Chat Completions

Model name*
gpt-3.5-turbo

☑ Use message payload

Message payload*
$messages

Model parameters

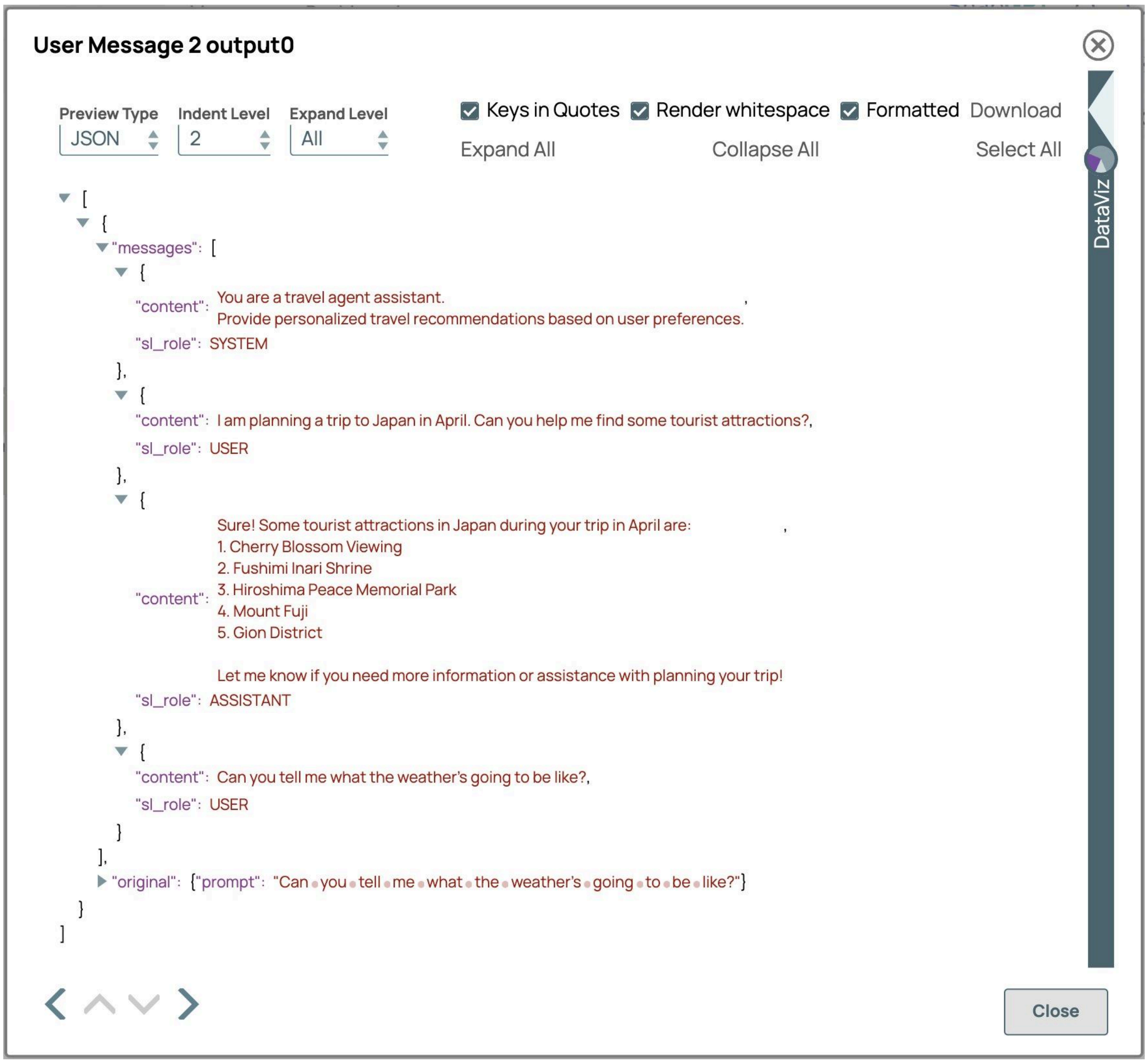
Advanced prompt configuration

Snap execution
Validate & Execute

31. Save and close the setting panel

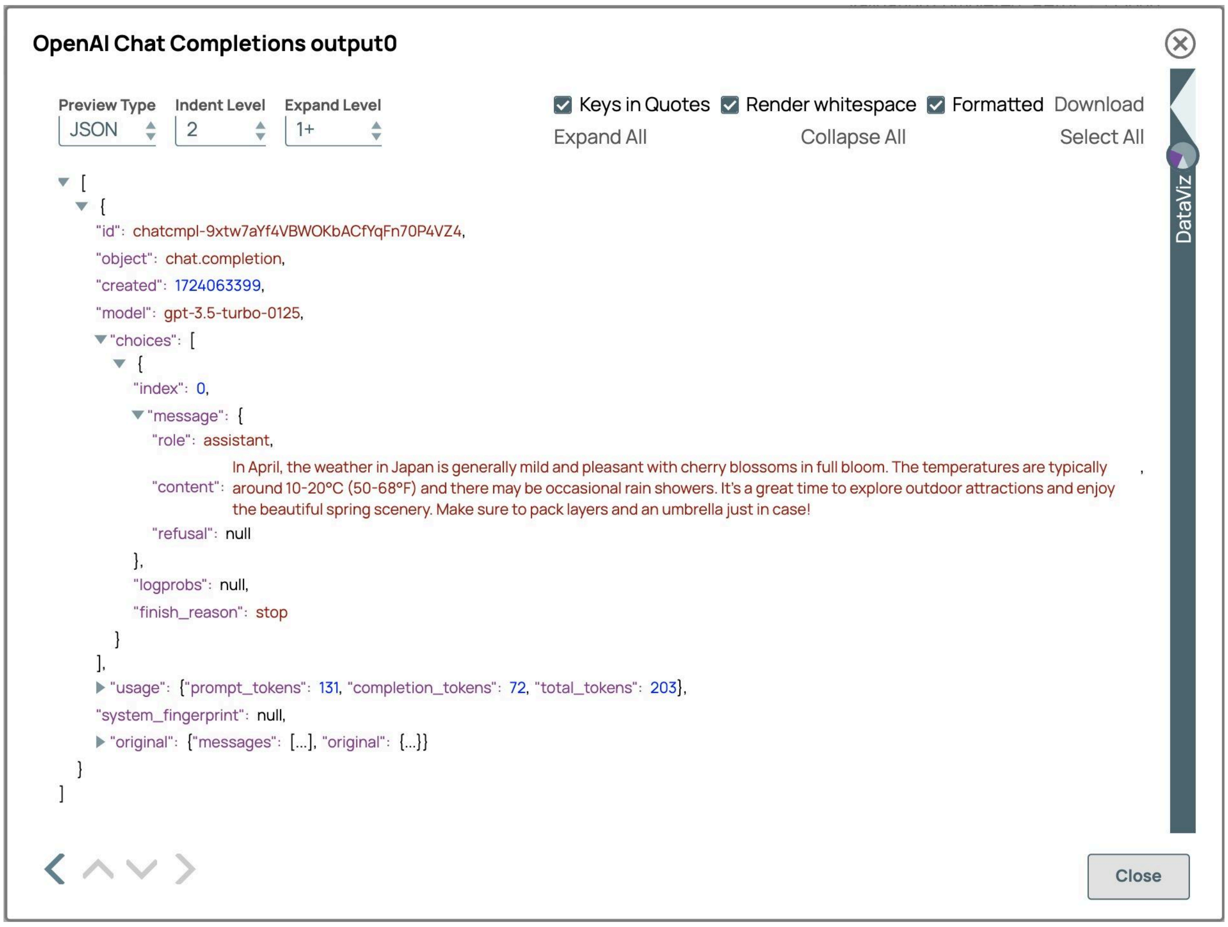
32. Validate the pipeline and let’s see the result.

32.1. Click on the output view of “User Message 2” to see the message payload, which we have constructed using the advanced mode of the prompt generator snap.



```
[
  {
    "messages": [
      {
        "content": You are a travel agent assistant.
                   Provide personalized travel recommendations based on user preferences.
        "sl_role": SYSTEM
      },
      {
        "content": I am planning a trip to Japan in April. Can you help me find some tourist attractions?,
        "sl_role": USER
      },
      {
        "content": Sure! Some tourist attractions in Japan during your trip in April are:
                   1. Cherry Blossom Viewing
                   2. Fushimi Inari Shrine
                   3. Hiroshima Peace Memorial Park
                   4. Mount Fuji
                   5. Gion District

                   Let me know if you need more information or assistance with planning your trip!
        "sl_role": ASSISTANT
      },
      {
        "content": Can you tell me what the weather's going to be like?,
        "sl_role": USER
      }
    ],
    "original": {"prompt": "Can you tell me what the weather's going to be like?"}
  }
]
```

32.2. Click on the output view of “Chat Completion” snap to see the LLM response.



```
[
  {
    "id": chatcmpl-9xtw7aYf4VBWOKbACfYqFn70P4VZ4,
    "object": chat.completion,
    "created": 1724063399,
    "model": gpt-3.5-turbo-0125,
    "choices": [
      {
        "index": 0,
        "message": {
          "role": assistant,
          "content": In April, the weather in Japan is generally mild and pleasant with cherry blossoms in full bloom. The temperatures are typically around 10-20°C (50-68°F) and there may be occasional rain showers. It's a great time to explore outdoor attractions and enjoy the beautiful spring scenery. Make sure to pack layers and an umbrella just in case!,
          "refusal": null
        },
        "logprobs": null,
        "finish_reason": stop
      }
    ],
    "usage": {"prompt_tokens": 131, "completion_tokens": 72, "total_tokens": 203},
    "system_fingerprint": null,
    "original": {"messages": [...], "original": {...}}
  }
]
```

The result is:

In April, the weather in Japan is generally mild and pleasant with cherry blossoms in full bloom. The temperatures are typically around 10-20°C (50-68°F) and there may be occasional rain showers. It's a great time to explore outdoor attractions and enjoy the beautiful spring scenery. Make sure to pack layers and an umbrella just in case!

The model effectively delivered weather information for Japan in April, even the last user query did not specify location or time. This is possible because the model uses the entire conversation history to understand the context and flow of the dialogue. Furthermore, the model echoed the user’s question before responding, maintaining a consistent conversational style. To achieve the best results, make sure your message list is complete and well-organized, as this will help the LLM generate more relevant and coherent responses, enhancing the quality of the interaction.

Tokens

Tokens are units of text, including words, character sets, or combinations of words and punctuation, that language models use to process and generate language. They can range from single characters or punctuation marks to entire words or parts of words, depending on the model. For instance, the word "artificial" might be split into tokens like "art", "ifi", and "cial". The total number of tokens in a prompt affects the model's response capability. Each model has a maximum token limit, which includes both the input and output. For instance, GPT-3.5-Turbo has a limit of 4,096 tokens, while GPT-4 has limits of 8,192 tokens and 32,768 tokens for the 32k context version. Effective token management ensures responses remain within these limits, improving efficiency, reducing costs, and enhancing accuracy.

To manage token usage effectively, the maximum tokens parameter is essential. It sets a limit on the number of tokens the model can generate, ensuring the combined total of input and output stays within the model's capacity. Setting a maximum tokens parameter has several benefits: it prevents responses from becoming excessively long, reduces response times by generating more concise outputs, optimizes performance, and minimizes costs by controlling token usage. Additionally, it enhances user experience by providing clear, focused, and quicker responses.

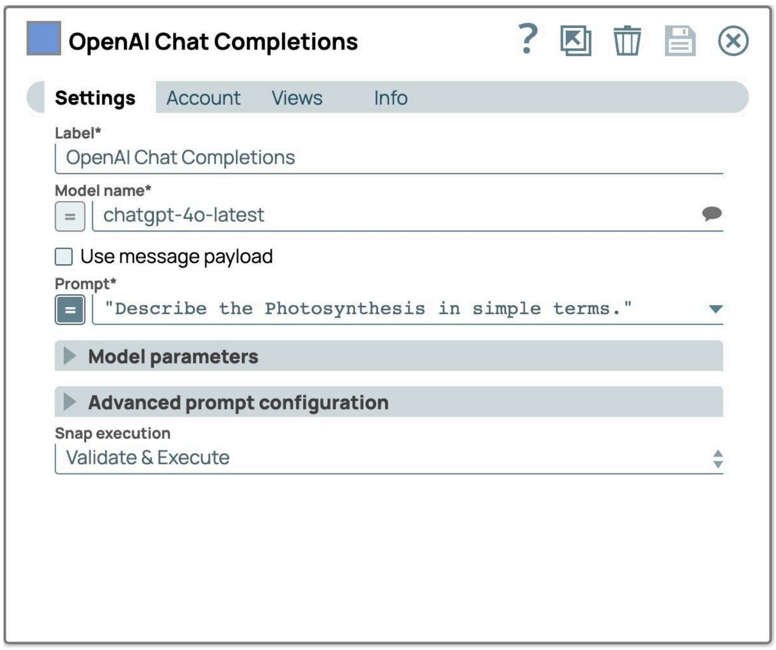
Use case Examples:

- **Customer Support Chatbots**: By setting maximum tokens, you ensure that the chatbot's responses are brief and focused, providing quick, relevant answers to user inquiries without overwhelming them with excessive detail. This enhances user experience and keeps interactions efficient.
- **Content summarization:** Helps generate concise summaries of long texts, suitable for applications with space constraints, such as mobile apps or notifications.
- **Interactive Storytelling:** Controls the length of narrative segments or dialogue options, maintaining engaging and well-paced storytelling.
- **Product Descriptions:** Generate brief and effective product descriptions for e-commerce platforms, maintaining relevance and fitting within space constraints.

Let's walk through how to configure the maximum tokens in the SnapLogic Chat Completion snap using the prompt: *"Describe the Photosynthesis in simple terms."*. We'll see how the LLM behaves with and without the maximum token setting.

1. Drag the "OpenAI Chat Completion" or "Azure OpenAI Chat Completion", or "Google Gemini Generate" onto the canvas
2. Select "Account" tab and select your configured account
3. Select "Settings" tab
4. Select your preferred model to use

5. Set prompt to the message *“Describe the Photosynthesis in simple terms.”*
   The Chat Completion settings should now look like this



Label*
OpenAI Chat Completions

Model name*
chatgpt-4o-latest

Use message payload

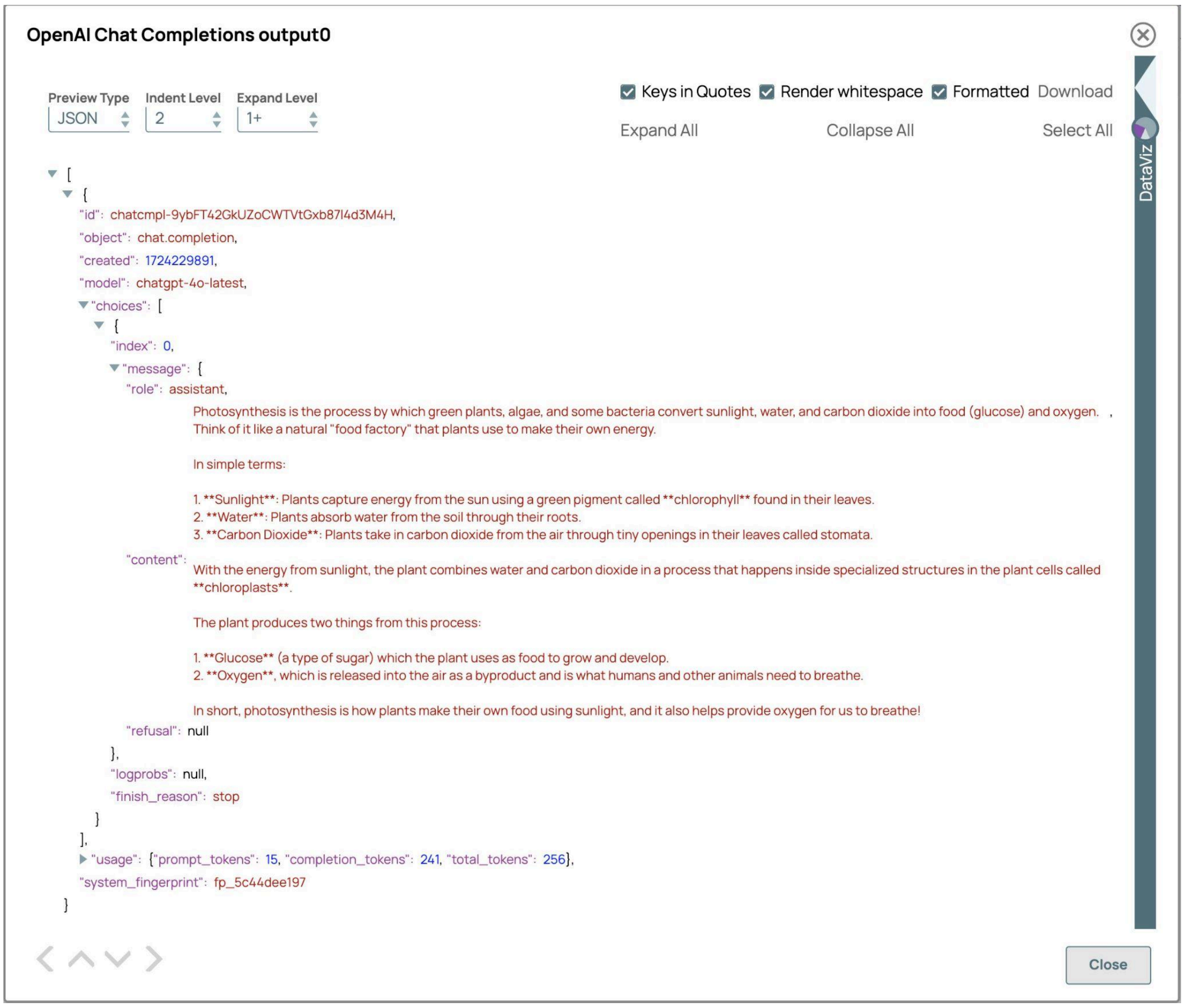
Prompt*
"Describe the Photosynthesis in simple terms."

Model parameters

Advanced prompt configuration

Snap execution
Validate & Execute

6. Save the snap settings and validate the pipeline to see the result.



Preview Type: JSON
Indent Level: 2
Expand Level: 1+

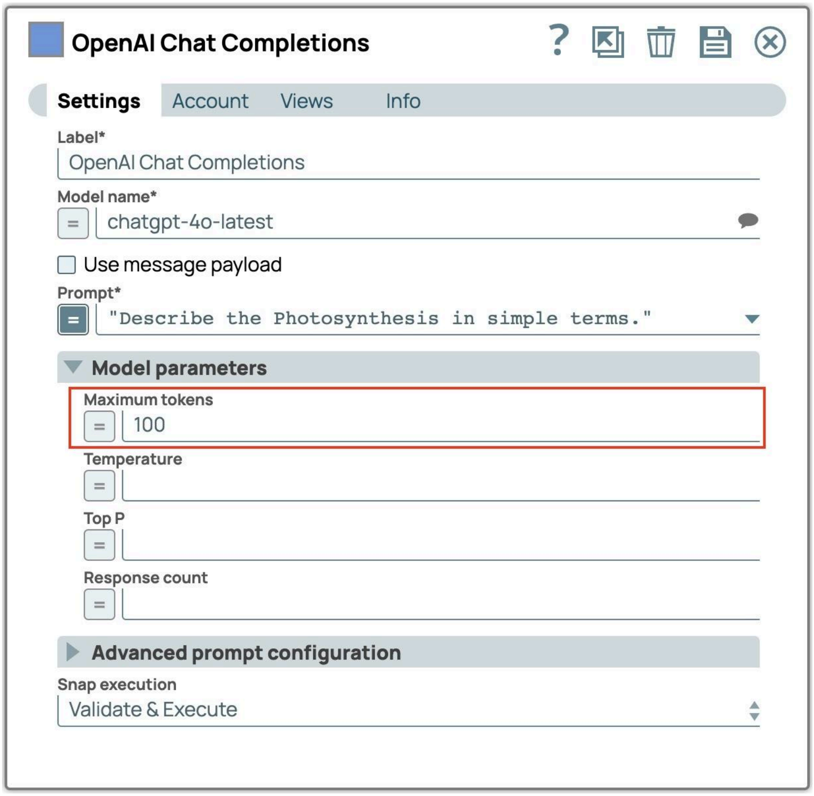
```
[
  {
    "id": chatcmpl-9ybFT42GkUZoCWTVtGxb87l4d3M4H,
    "object": chat.completion,
    "created": 1724229891,
    "model": chatgpt-4o-latest,
    "choices": [
      {
        "index": 0,
        "message": {
          "role": assistant,
                    Photosynthesis is the process by which green plants, algae, and some bacteria convert sunlight, water, and carbon dioxide into food (glucose) and oxygen.  ,
                    Think of it like a natural "food factory" that plants use to make their own energy.

                    In simple terms:

                    1. **Sunlight**: Plants capture energy from the sun using a green pigment called **chlorophyll** found in their leaves.
                    2. **Water**: Plants absorb water from the soil through their roots.
                    3. **Carbon Dioxide**: Plants take in carbon dioxide from the air through tiny openings in their leaves called stomata.
          "content":
                    With the energy from sunlight, the plant combines water and carbon dioxide in a process that happens inside specialized structures in the plant cells called
                    **chloroplasts**.

                    The plant produces two things from this process:

                    1. **Glucose** (a type of sugar) which the plant uses as food to grow and develop.
                    2. **Oxygen**, which is released into the air as a byproduct and is what humans and other animals need to breathe.

                    In short, photosynthesis is how plants make their own food using sunlight, and it also helps provide oxygen for us to breathe!
          "refusal": null
        },
        "logprobs": null,
        "finish_reason": stop
      }
    ],
    "usage": {"prompt_tokens": 15, "completion_tokens": 241, "total_tokens": 256},
    "system_fingerprint": fp_5c44dee197
  }
```

In the result, the “usage” field provide us the token consumption detail.

- prompt_tokens: tokens used by the input
- completion_tokens: tokens used for generating response.
- total_tokens: the combined number of tokens used for both the input prompt and the generated response

We can see that the response is quite long and the token used for response(completion_tokens) is 241. Let’s set the maximum token and see the result again

7. Expand the “Model parameters”
8. Set “Maximum tokens” to 100



Label*
OpenAI Chat Completions

Model name*
chatgpt-4o-latest

Use message payload

Prompt*
"Describe the Photosynthesis in simple terms."

Maximum tokens
100

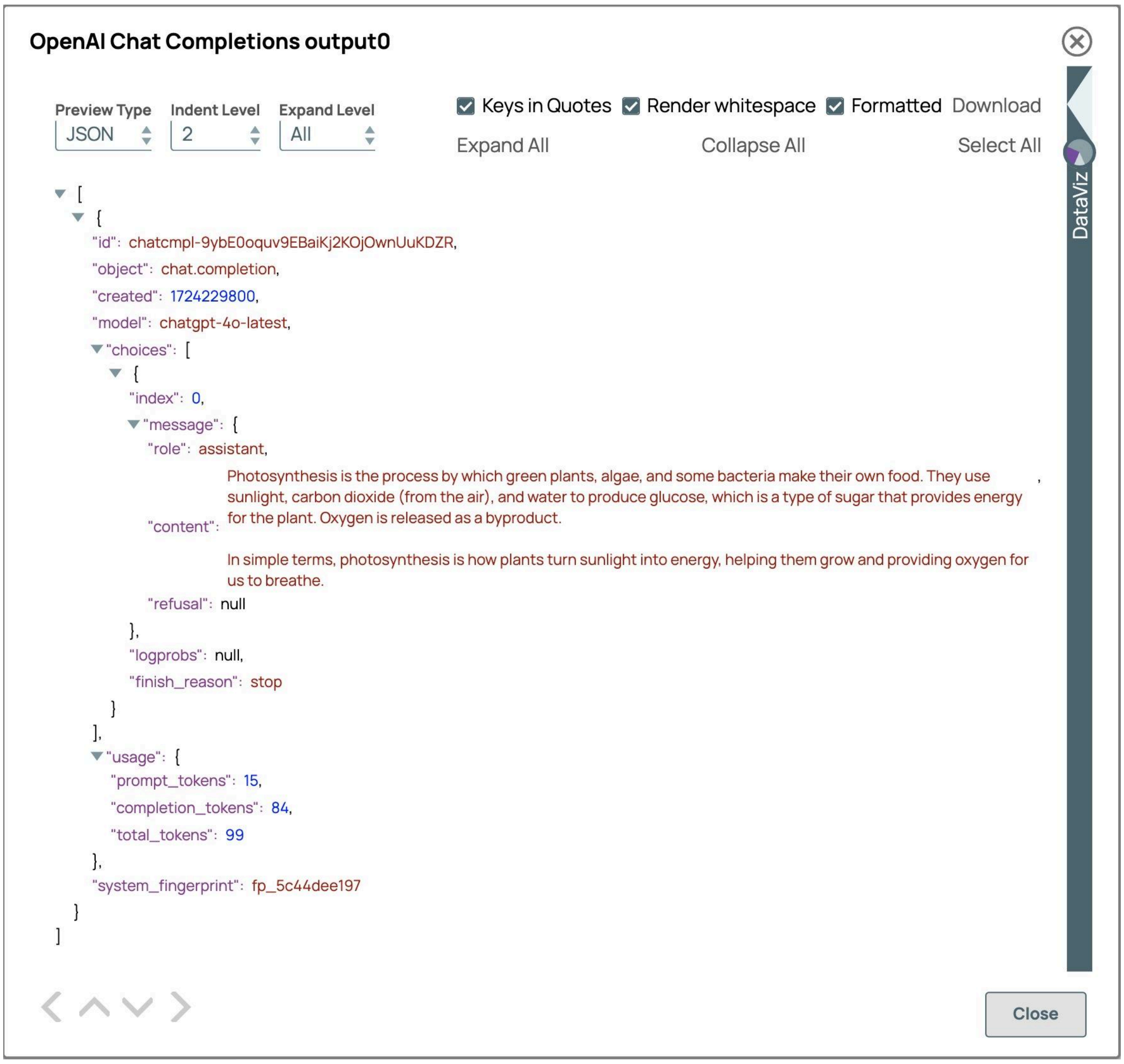
Temperature

Top P

Response count

Snap execution
Validate & Execute

9. Save the snap settings and validate the pipeline to see the result.



```
[
  {
    "id": chatcmpl-9ybE0oquv9EBaiKj2KOjOwnUuKDZR,
    "object": chat.completion,
    "created": 1724229800,
    "model": chatgpt-4o-latest,
    "choices": [
      {
        "index": 0,
        "message": {
          "role": assistant,
          "content": Photosynthesis is the process by which green plants, algae, and some bacteria make their own food. They use sunlight, carbon dioxide (from the air), and water to produce glucose, which is a type of sugar that provides energy for the plant. Oxygen is released as a byproduct.

                     In simple terms, photosynthesis is how plants turn sunlight into energy, helping them grow and providing oxygen for us to breathe.,
          "refusal": null
        },
        "logprobs": null,
        "finish_reason": stop
      }
    ],
    "usage": {
      "prompt_tokens": 15,
      "completion_tokens": 84,
      "total_tokens": 99
    },
    "system_fingerprint": fp_5c44dee197
  }
]
```

The result is more concise compared to the output when the maximum tokens are not set. In this case, the number of completion_tokens used is only 84, indicating a shorter and more focused response.

Using the maximum tokens effectively ensures that responses are concise and relevant, optimizing both performance and cost-efficiency. By setting this limit, you prevent excessively long outputs, reduce response times, and maintain clarity in the generated content. To achieve optimal results, align the maximum tokens setting with your specific needs, such as the desired response length and application requirements. Regularly review and adjust this parameter to balance brevity with completeness, ensuring that the outputs remain useful and within operational constraints.

Prompt size considerations

In the previous section, we covered techniques for managing response size to stay within token limits. Now, we turn our focus to prompt size and context considerations. By ensuring that both prompts and context are appropriately sized, you can improve the accuracy and relevance of model responses while staying within token limits.

Here are some techniques for managing prompt and context size:

- **Keep prompt clear and concise**
  By making prompts clear and direct, you reduce token usage, which helps keep the prompt within the model's limits. Focusing on essential information and removing unnecessary words enhances the accuracy and relevance of the model's responses. Additionally, specifying the desired output length further optimizes the interaction, preventing excessively long responses and improving overall efficiency.

  Example
  Prompt: *“Could you please provide a detailed explanation of how the process of photosynthesis works in plants, including the roles of chlorophyll, sunlight, and water?”*
  Better prompt: *"Explain the process of photosynthesis in plants, including the roles of chlorophyll, sunlight, and water, in about 50 words."*

- **Splitting complex tasks into simpler prompts**
  Breaking down complex tasks into smaller, more manageable subtasks not only reduces the size of each individual prompt but also enables the model to process each part more efficiently. This approach ensures that each prompt stays within token limits, resulting in clearer and more accurate responses.

  Example
  Complex Task:
  "Write a detailed report on the economic impact of climate change in developing countries, including statistical analysis, case studies, and policy recommendations."

  Simplified Prompts:
  1. *"Summarize the economic impact of climate change in developing countries."*
  2. *"Provide a statistical analysis of how climate change affects agriculture in developing countries."*
  3. *"List case studies that demonstrate the economic consequences of climate change in developing countries."*
  4. *"Suggest policy recommendations for mitigating the economic impact of climate change in developing countries."*

- **Use a sliding window for chat history**
  From the complex prompt section, we know that including the entire chat history helps maintain context, but it can also quickly use up available tokens. To optimize prompt size, employ a sliding window approach. This technique involves including only a portion of the chat history, focusing on recent and relevant exchanges, to keep the prompt within token limits.

- **Summarize contexts**
  Use a summarization technique to condense context into a brief summary. Instead of including extensive conversation history, create a concise summary that captures the essential information. This approach reduces token usage while retaining key details for generating accurate responses.

By applying these techniques, you can effectively manage prompt and context size, ensuring that interactions remain efficient and relevant while optimizing token usage.