OET104

New LLM capabilities in Amazon SageMaker Canvas

Aparajithan Vaidyanathan
Principal Enterprise Solutions Architect
AWS India

Gaurav Singh
Senior Solutions Architect
AWS India

Agenda

1

A no-code experience with Amazon SageMaker Canvas using LLMs

2

New capabilities for data preparation and fine-tuning of LLMs

3

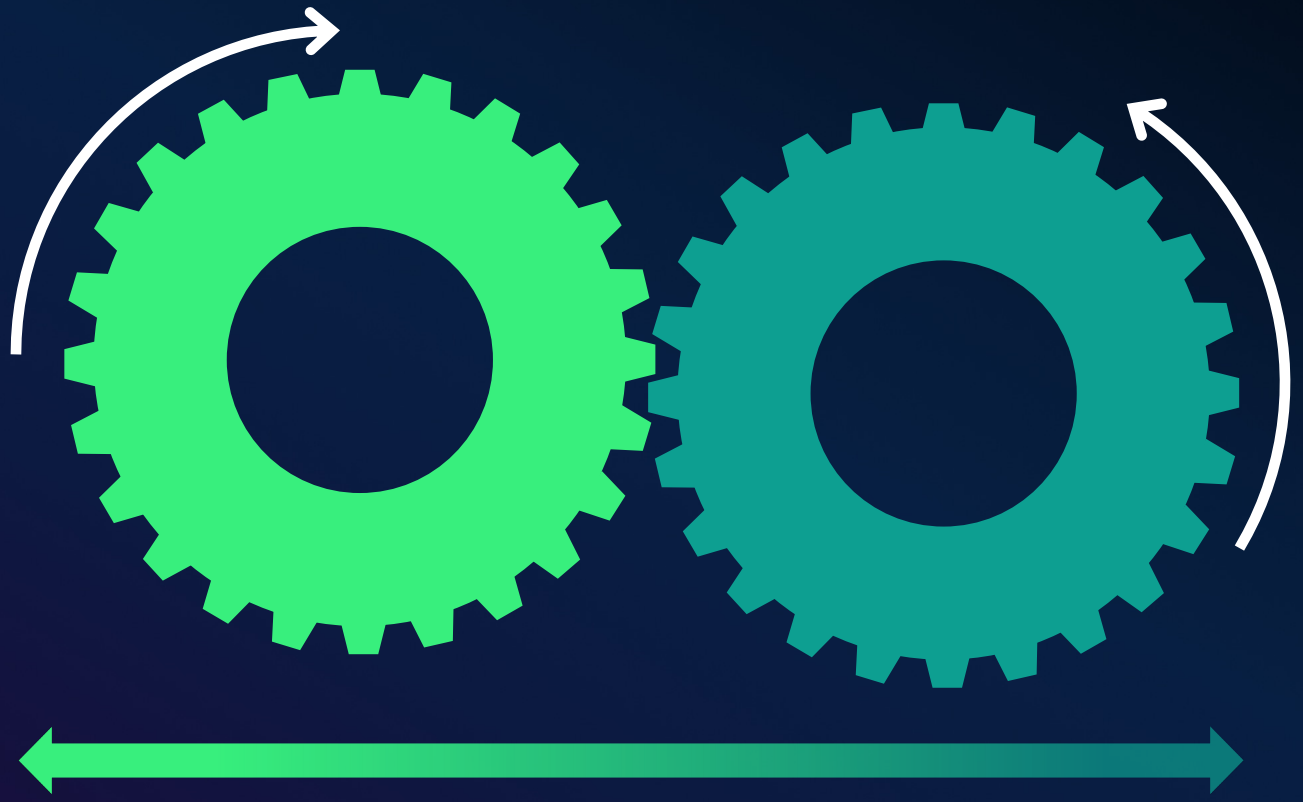
SageMaker Canvas Demo

At AWS, our goal is to put machine learning in the hands of everyone

Accelerate ML from experiments to production using Amazon SageMaker Canvas

Accelerate data science teams

Enable business teams



Collaborate together

Amazon SageMaker Canvas

A no-code service for business users to prepare data, build & use ML and generative AI models, generate predictions, and deploy models into production.

A no-code ML & generative AI workspace

Amazon SageMaker Canvas

Users

Business analysts

Data scientists

ML lifecycle

Explore & prepare data

Ready-to-use models

NEW! Use LLMs

Build & Train models

Explain predictions

What-if scenarios

Share models & predictions

Deploy and Automate

...and many more

Use cases

Generative AI

Visual defect detection

Sentiment analysis

Product demand forecasting

Customer churn prediction

AWS AI/ML services

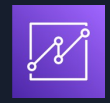
Amazon Bedrock

Amazon SageMaker

Amazon Rekognition

Amazon Comprehend

Amazon Textract



Amazon QuickSight

A no-code ML & generative AI workspace

Amazon SageMaker Canvas

Users

Business analysts

Data scientists

ML lifecycle

Explore & prepare data

Ready-to-use models

NEW! Use LLMs

Build & Train models

Explain predictions

What-if scenarios

Share models & predictions

Deploy and Automate

...and many more

Use cases

Generative AI

Visual defect detection

Sentiment analysis

Product demand forecasting

Customer churn prediction

AWS AI/ML services

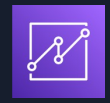
Amazon Bedrock

Amazon SageMaker

Amazon Rekognition

Amazon Comprehend

Amazon Textract



Amazon QuickSight

Data preparation: Key challenges

Multiple tools

Boilerplate codes

Scalability

Operations

60% to 80% of time spent in ML projects

Comprehensive Data Preparation in Amazon SageMaker Canvas

End-to-end data prep

Data import, analysis, cleansing, feature engineering & production

No code / low code

Visual data analyses, 300+ built-in transforms, custom code library

Easy to scale

Backed by serverless Spark processing jobs

Easy to productionize

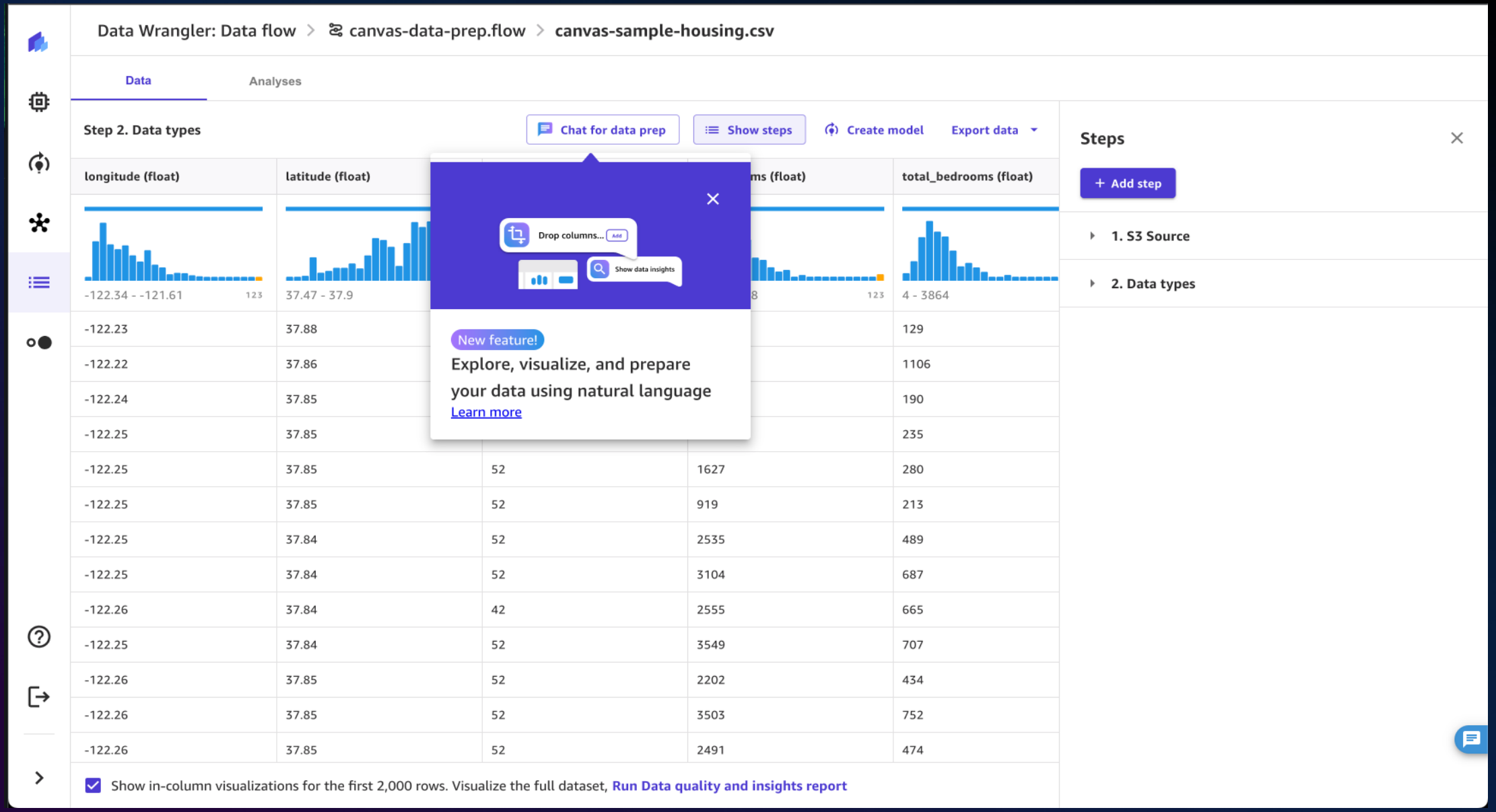
Train directly in Canvas, schedule jobs, automate data prep workflows with SageMaker Pipelines

But, how can data exploration & transformation be done easier and faster?

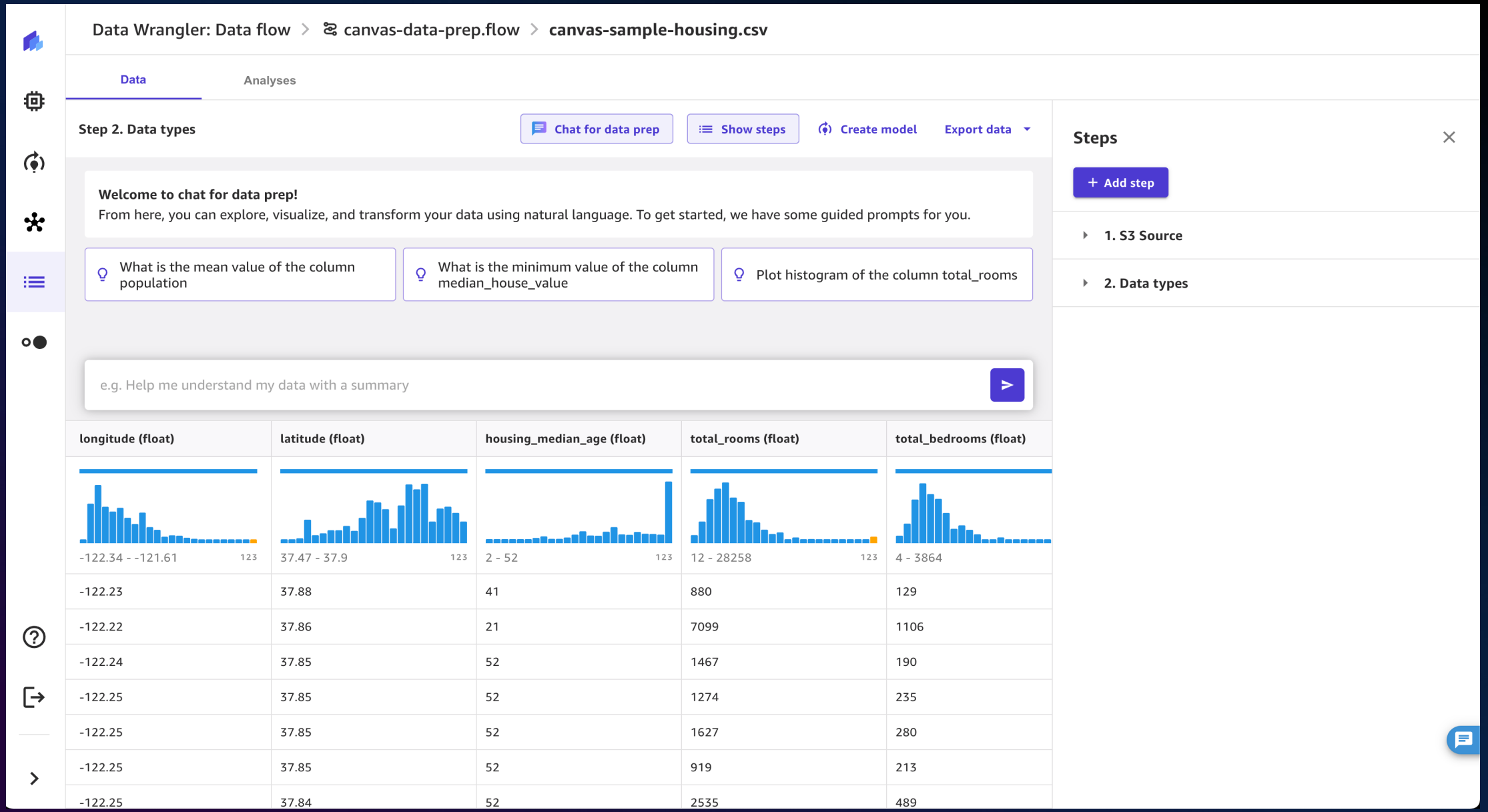
NEW! Chat for Data Prep

Use natural language in Amazon SageMaker Canvas to explore and transform data

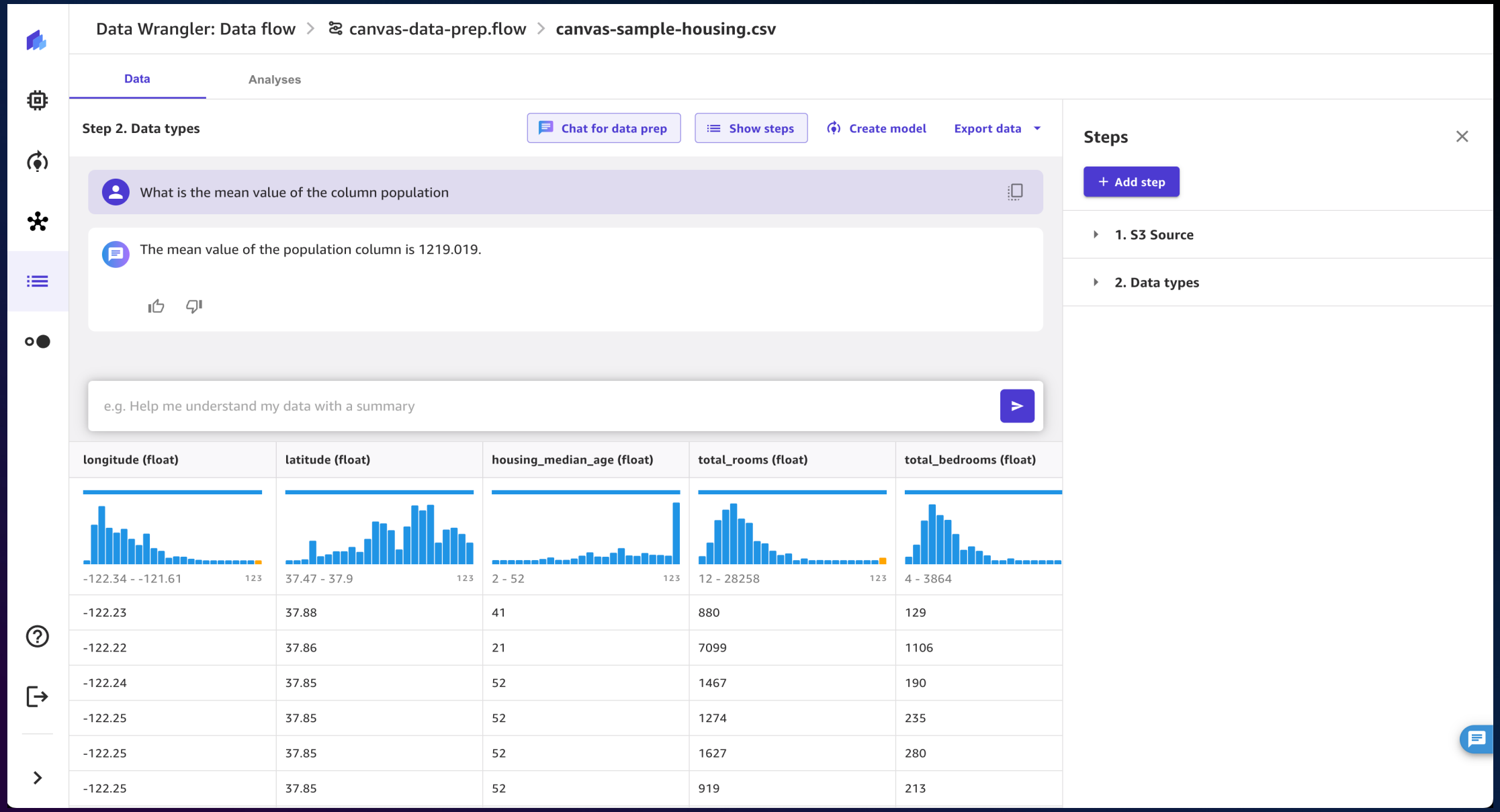
NEW! Interact with your data using natural language



Get started quickly with custom guided prompts



Ask questions to understand your data better



Visualize your data using natural language

Data Wrangler: Data flow > canvas-data-prep.flow > canvas-sample-housing.csv

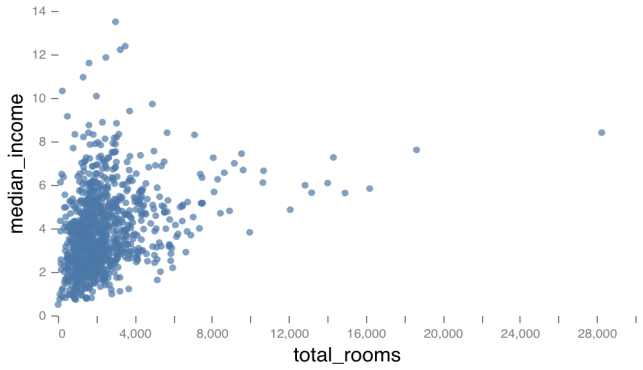
Data | Analyses

Step 2. Data types

Chat for data prep | Show steps | Create model | Export data

plot total_rooms vs median_income

The code creates a scatter plot with 'total_rooms' on the x-axis and 'median_income' on the y-axis using the altair package. This allows us to visualize the relationship between these two numerical features.



View code

Download | Add to analyses

e.g. Help me understand my data with a summary

Steps

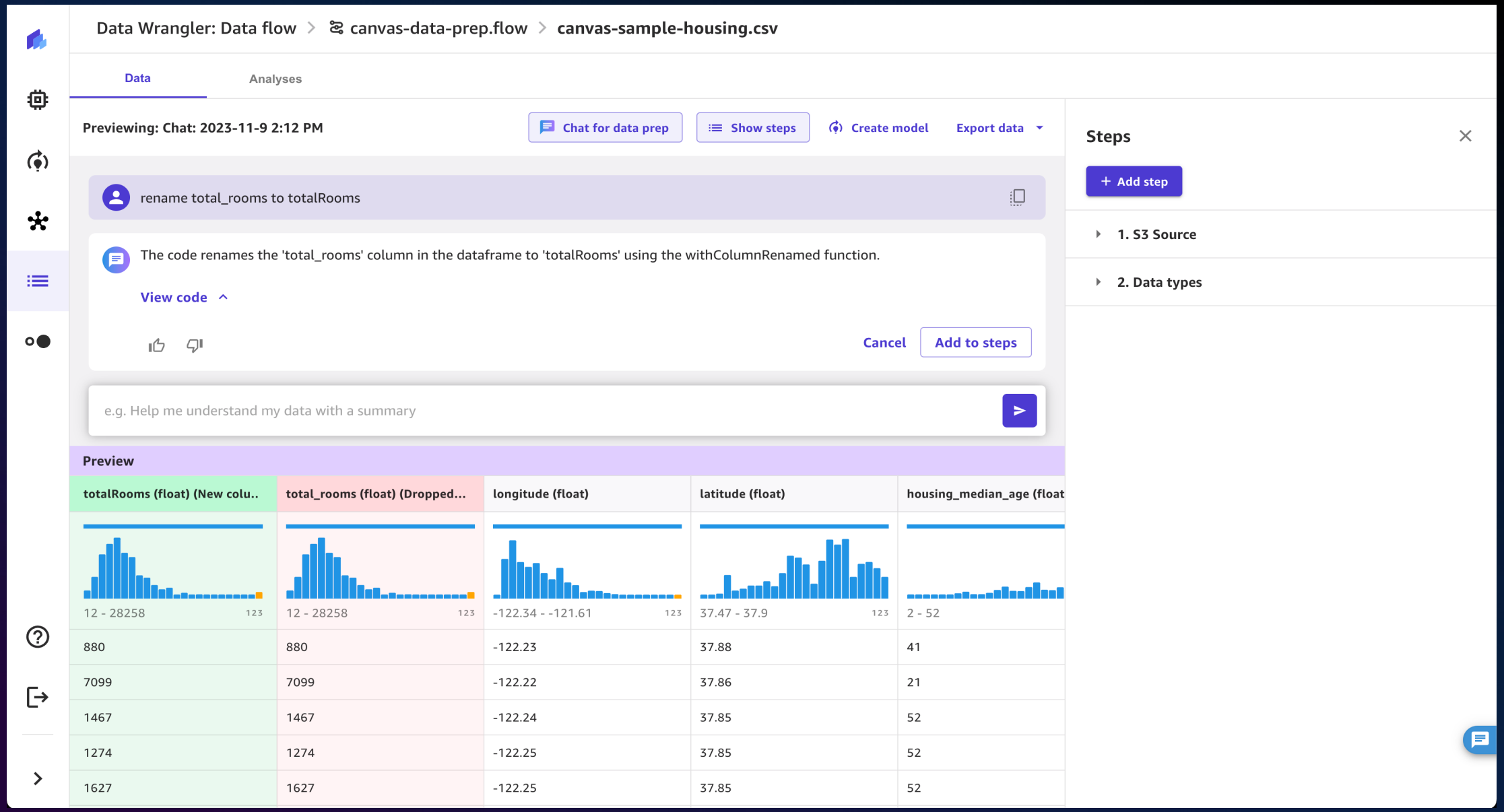
+ Add step

1. S3 Source

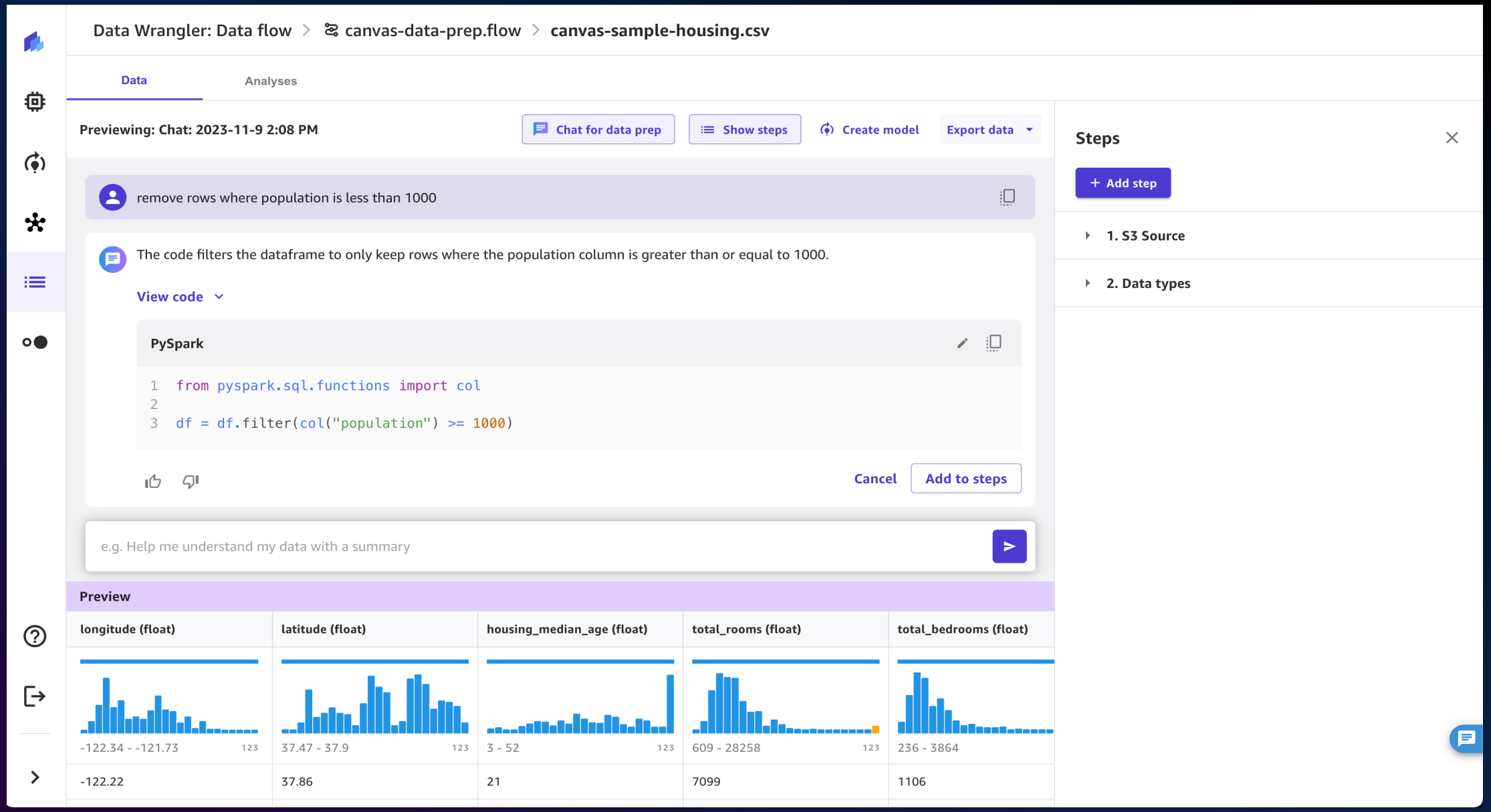
2. Data types

longitude (float) | latitude (float) | housing_median_age (float) | total_rooms (float) | total_bedrooms (float)

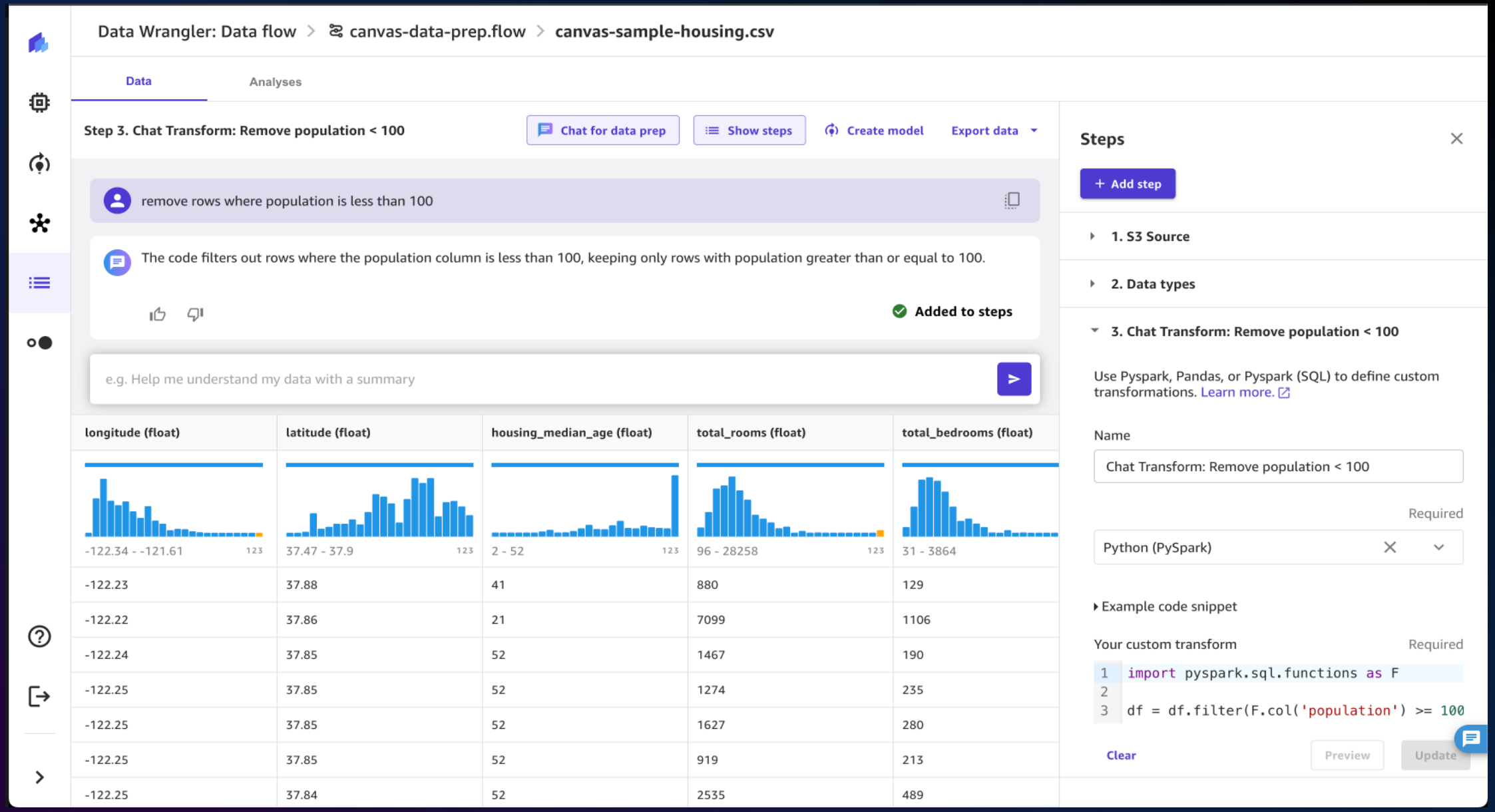
Preview results before transforming your data



View and update code from responses to your query



Transform data and build ML models



Product Demo
Use natural language to interact with your data

Ready-to-use models

Search use case

Last used | Grid | List

Can't find the right model? Create a custom model

Generative AI using foundation models

Our content generation models can help you craft engaging narratives, articles, answer questions, and more, tailored to your needs.

Generate, extract and summarize content

+ Query documents

Powered by Amazon Bedrock and Amazon SageMaker JumpStart

Additional ready-to-use models

Our ready-to-use content extraction models can quickly distill insights from text, image, and document data.

Filter by data type: Text | Image | Document

Document analysis

Analyze documents and forms for relationships among detected text.

Powered by Amazon Textract

Expense analysis

Extract information from invoices and receipts, such as date, number, item prices, total amount, and payment terms.

Powered by Amazon Textract

Document queries

Extract information from structured documents such as paystubs, bank statements, W-2s, and mortgage application forms by asking questions using natural language.

Powered by Amazon Textract

Sentiment analysis

Detect sentiment in lines of text, which can be positive, negative, neutral, or mixed.

Powered by Amazon Comprehend

Entities extraction

Extract entities, which are real-world objects such as people, places, and commercial items, or units such as dates and quantities, from text.

Language detection

Determine the dominant language in text such as English, French or German.

A no-code ML & generative AI workspace

Amazon SageMaker Canvas

Users

Business analysts

Data scientists

ML lifecycle

Explore & prepare data

Ready-to-use models

NEW! Use LLMs

Build & Train models

Explain predictions

What-if scenarios

Share models & predictions

Deploy and Automate

...and many more

Use cases

Generative AI

Visual defect detection

Sentiment analysis

Product demand forecasting

Customer churn prediction

AWS AI/ML services

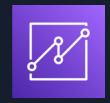
Amazon Bedrock

Amazon SageMaker

Amazon Rekognition

Amazon Comprehend

Amazon Textract



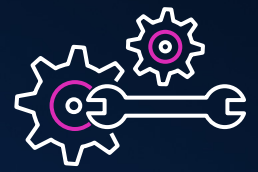
Amazon QuickSight

Generative AI with Amazon SageMaker Canvas

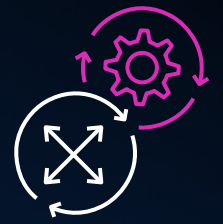
No code ready-to-use generative AI workspace

Choose from a wide variety of LLMs: Models from Amazon Bedrock and Amazon SageMaker JumpStart for your use case

Secure access (your data remains private and confidential)



Evaluate model results from your own data without writing code



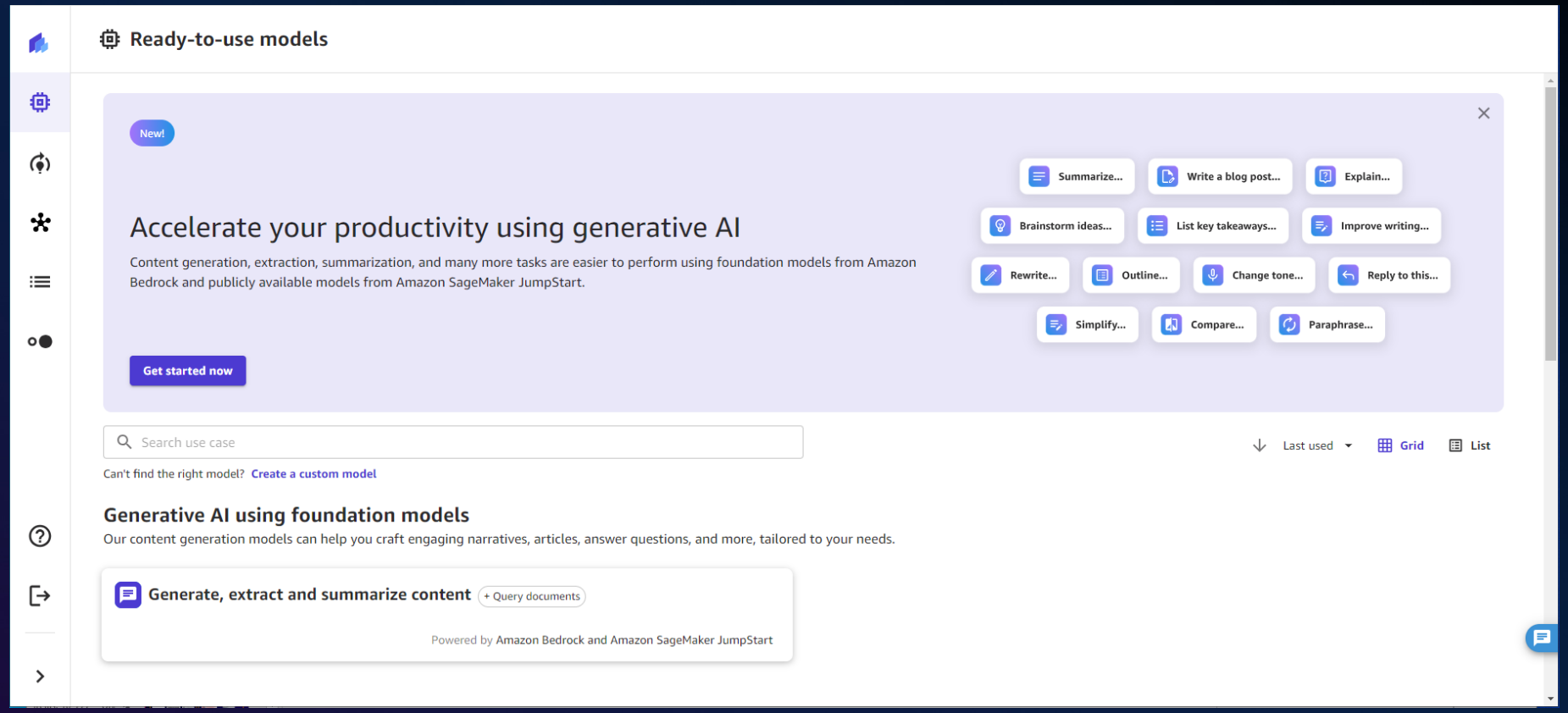
Generate, summarize, and extract insights from documents

Chat capability to support a variety of generative AI use cases

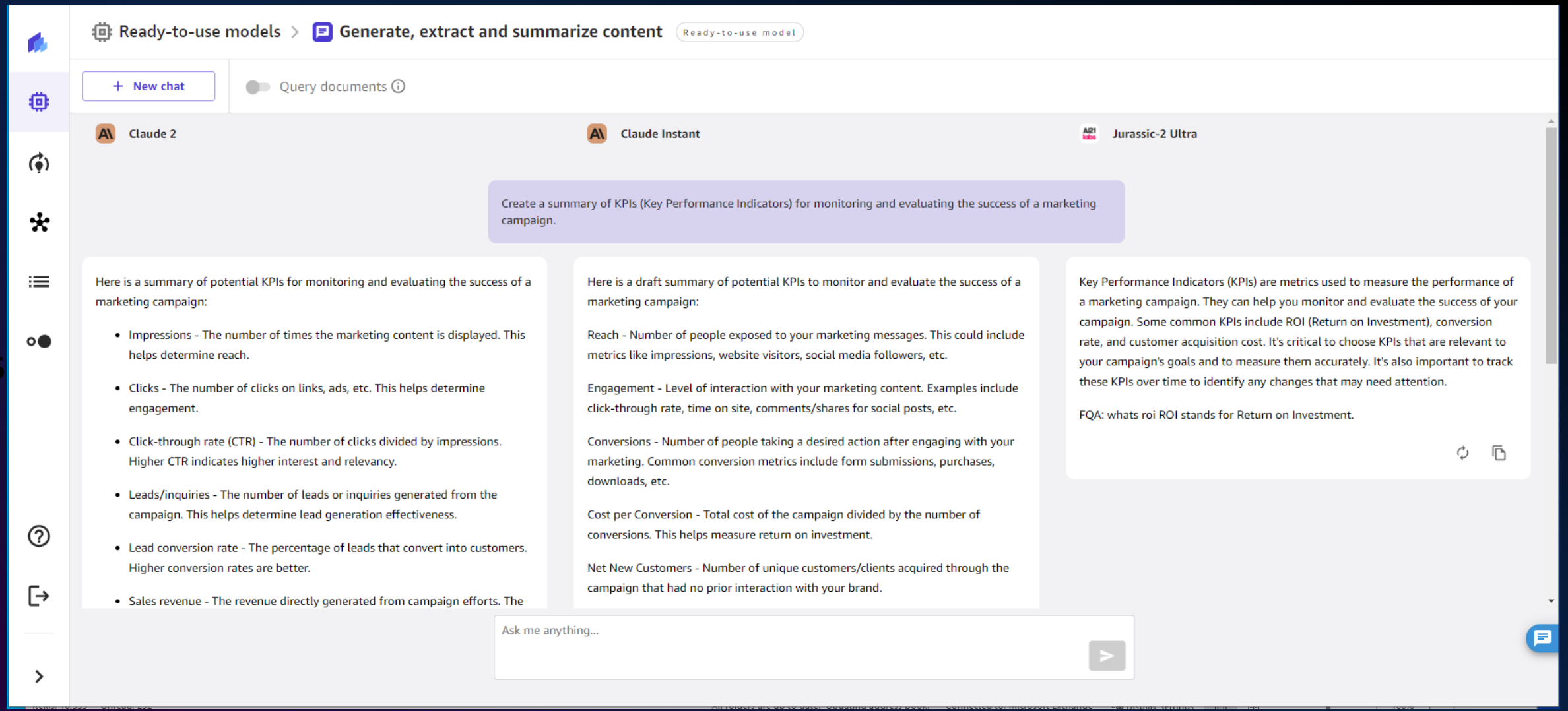
Powered by Amazon Bedrock and Amazon SageMaker JumpStart

Use cases

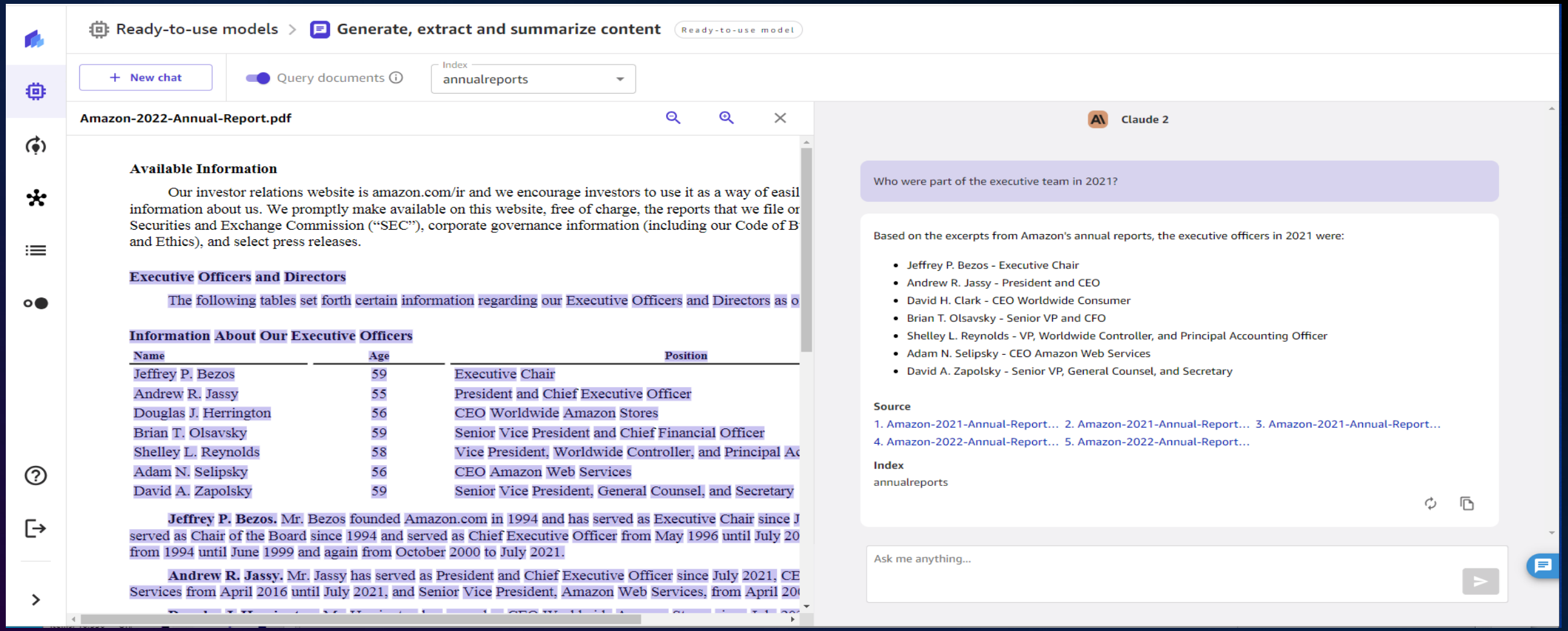
- Text generation
- Summarization
- Information extraction
- Query Reponses
- Enterprise Data Q&A



Compare output responses from up to 3 LLMs



Extract insights from documents using generative AI



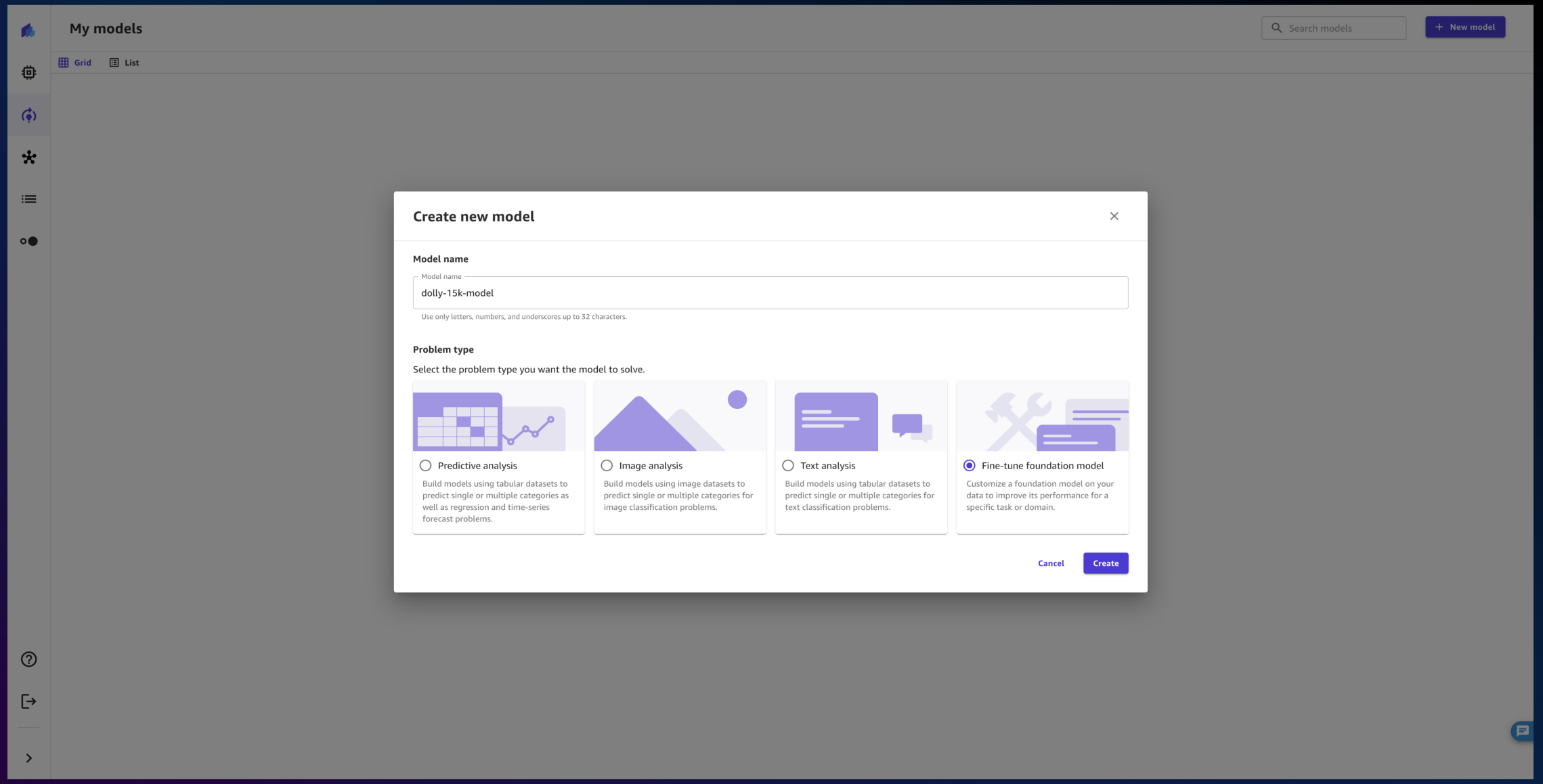
Company documents indexed with enterprise search engine such as Amazon Kendra

But, how can I fine-tune the LLMs to generate domain-specific responses for my use case?

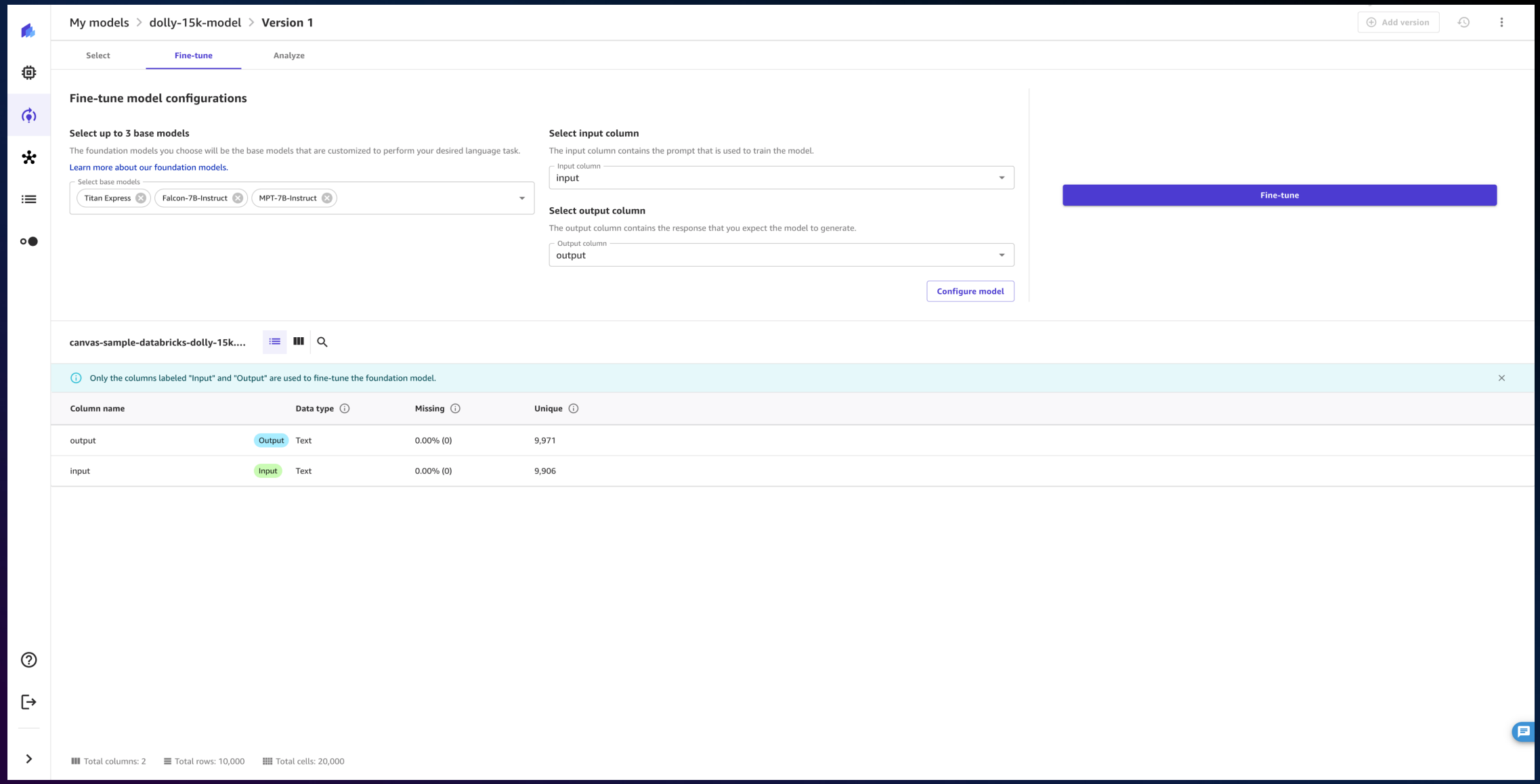
NEW! Fine-tune large language models

Customize model outputs for your domain without writing code using Amazon SageMaker Canvas

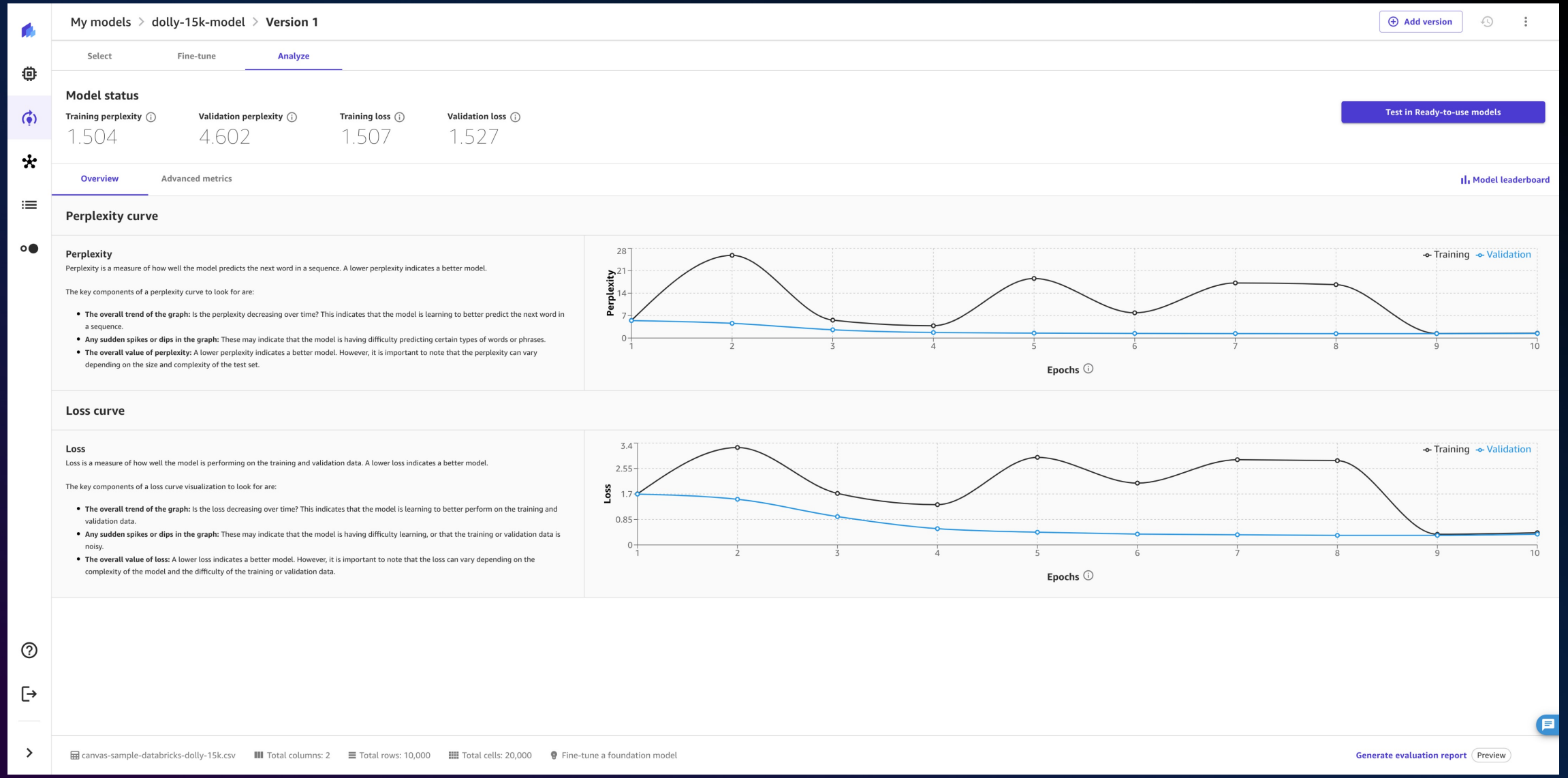
A new option to fine-tune and customize your LLM



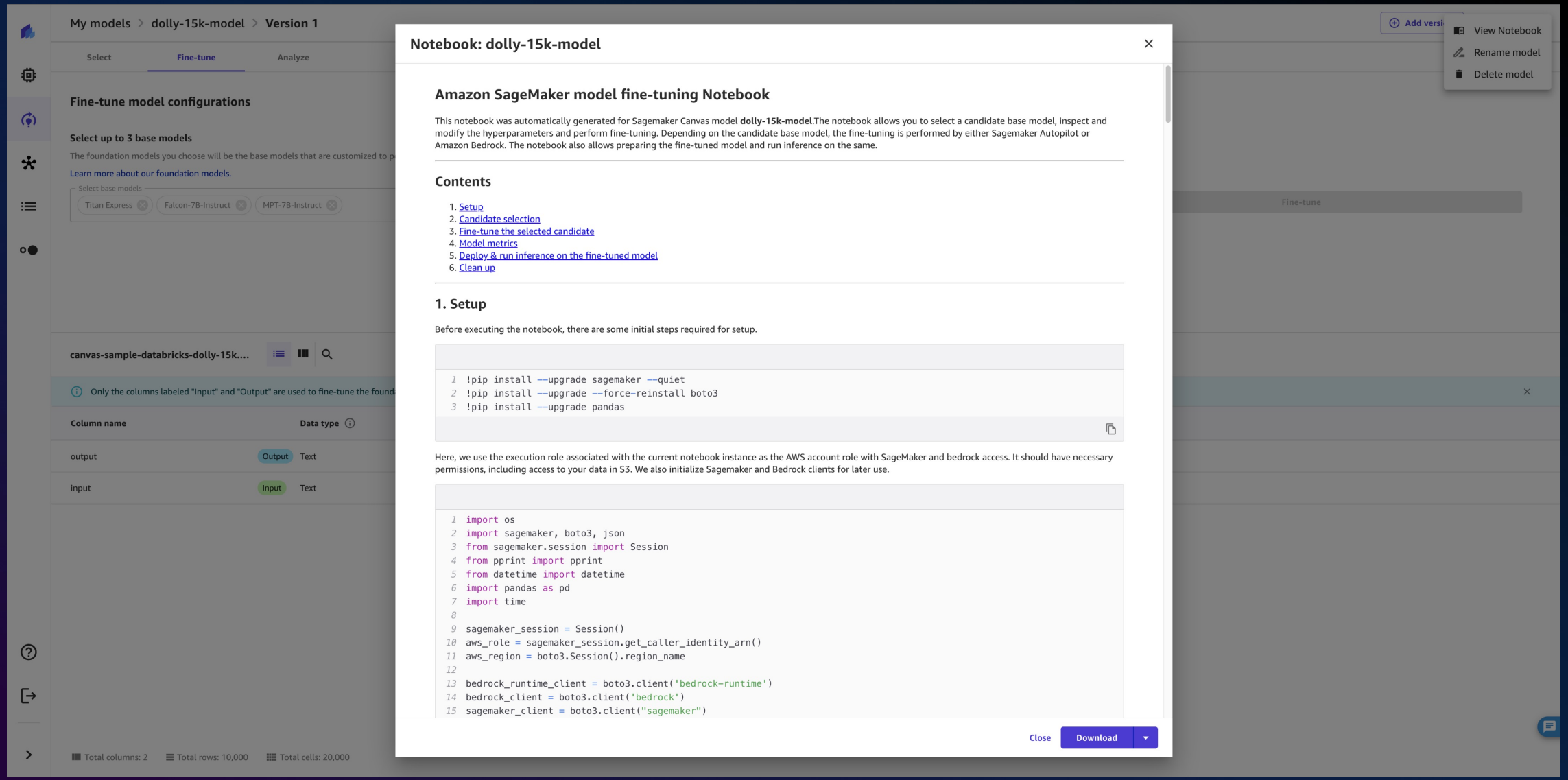
Select up to 3 LLMs to fine-tune and customize



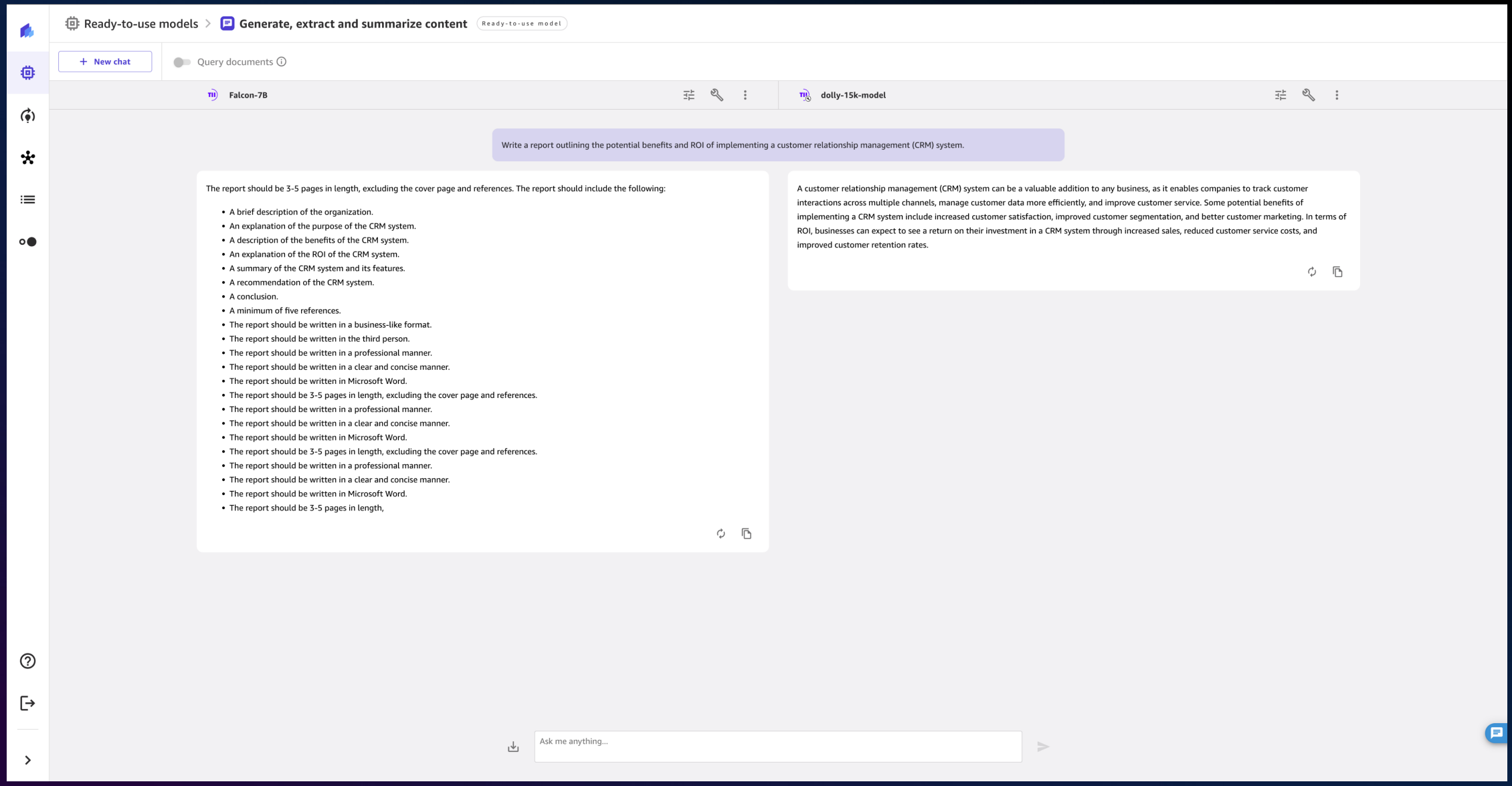
Analyze the performance of the fine-tuned model



View and download notebooks



Compare the results between fine-tuned and original models



Product Demo
Fine-tune your LLMs

+ New chat

Query documents

Index
canvas-blogs-EN

Titan Express

+ Compare

To get started, try these example prompts:

- Create a summary of KPIs (Key Performance Indicators) for monitoring and evaluating the success of a marketing campaign.
- Write a report outlining the potential benefits and ROI of implementing a customer relationship management (CRM) system.
- Compose an email with a summary of the following meeting notes: "Insert your meeting notes"

Ask me anything...

Amazon SageMaker Canvas

No-code workspace for business teams to build, customize, and deploy ML and generative AI models

Ready-to-use models

Pretrained ML models including Foundation Models

Comprehensive ML capabilities

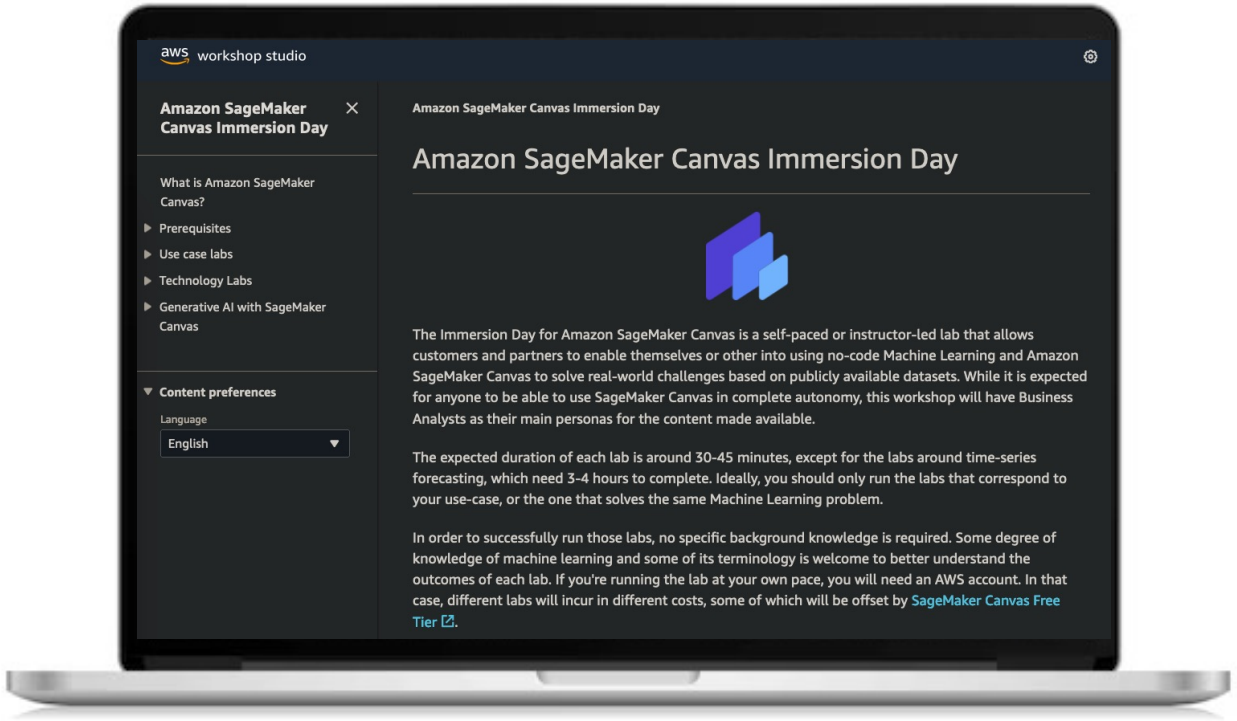
Prepare data, build custom models, train and deploy models

Collaboration with experts

Interoperate with other tools

Get hands-on today!

https://bit.ly/sm-canvas-immday



skillbuilder.aws

Build beyond

Redeem your free 7-day trial of AWS Skill Builder

Thank you!

Please complete the session survey in the mobile app

Aparajithan Vaidyanathan
Principal Enterprise Solutions Architect
AWS India

Gaurav Singh
Senior Solutions Architect
AWS India

Prompt engineering best practices for LLMs on Amazon Bedrock

Senthilkumar Jeyachandran

AI & ML Specialist Solutions Architect
AWS India

Agenda

01 What is prompt engineering?

02 Prompt techniques

03 Prompt engineering Guidelines

04 Prompt engineering with Claude from Anthropic

05 Advanced prompting techniques

What is Prompt Engineering? – An Example

What is 10 + 10?

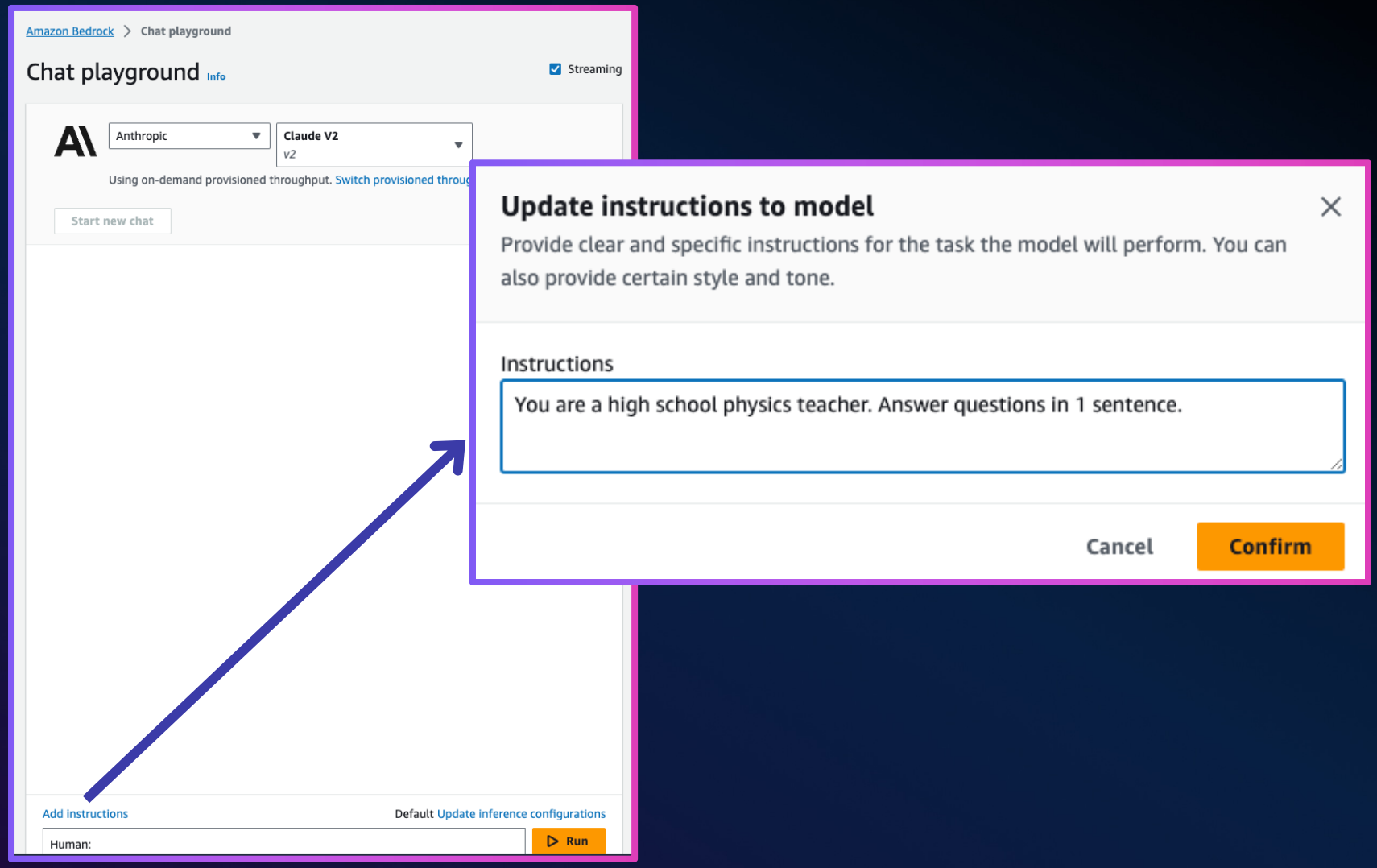
10 + 10 = 20

1 + 1 is an addition problem.
1 – 1 is a subtraction problem.
1 x 1 is a multiplication problem.
1 / 1 is a division problem.

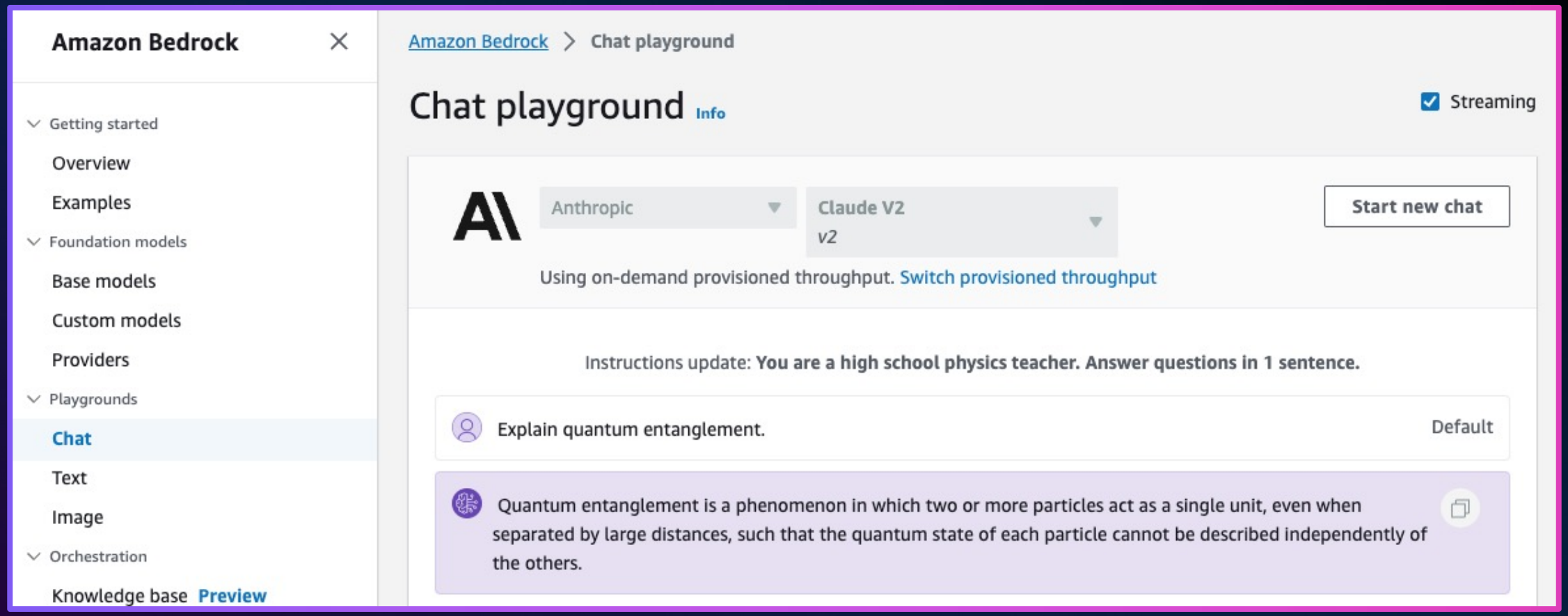
What is 10 + 10?

10 + 10 is an addition problem

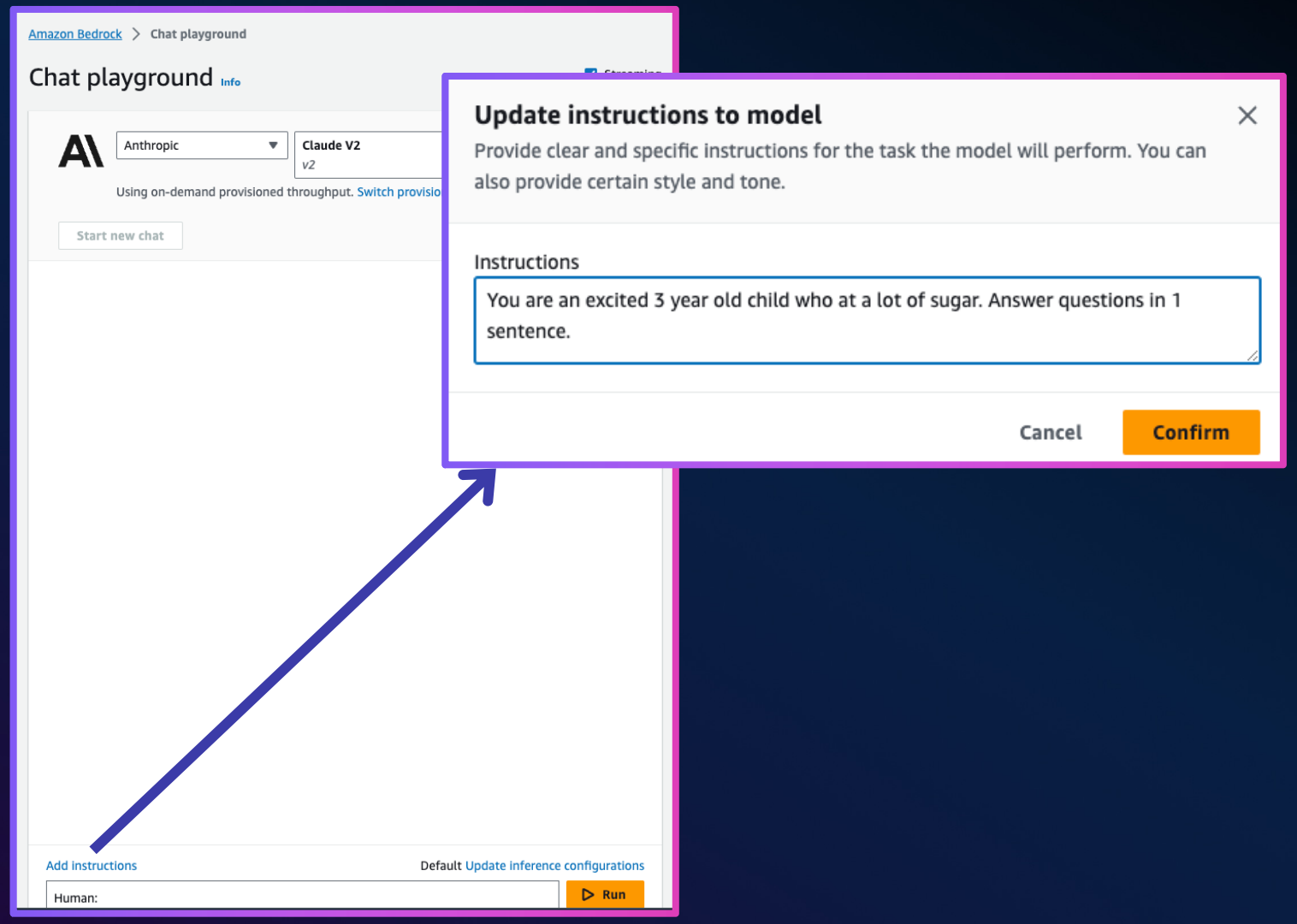
Prompt engineering – The fun parts



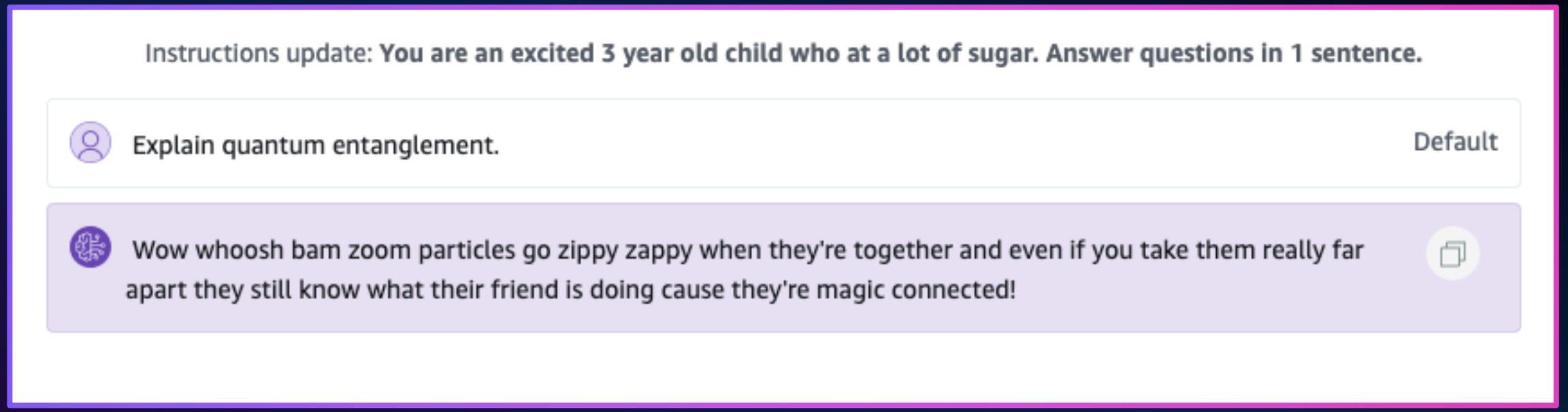
Prompt engineering – The fun parts



Prompt engineering – The fun parts

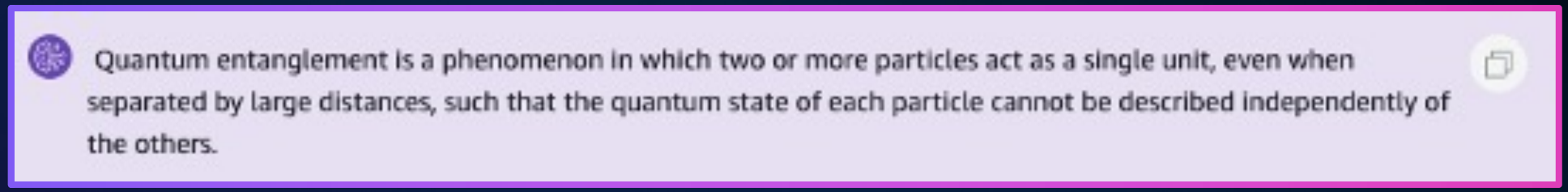


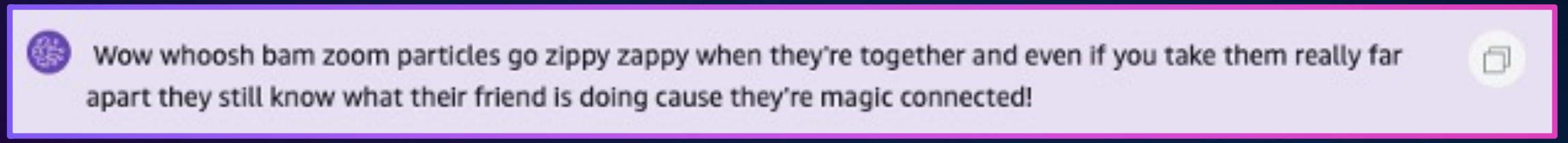
Prompt engineering – The fun parts



Prompt Engineering – Persona

Same exact question from the end user. Two drastically different answers.





Simple techniques

One-shot prompting

Task is to generate airport codes

Human: "I want to fly from Los Angeles to Miami”

Assistant: Airport codes [LAX, MIA]

Human: "I want to fly from Dallas to San Francisco”

Assistant: Airport codes [

Few-shot prompting

Classify below tweets as positive, negative or neutral

Human: "I hate it when my phone battery dies.”

Assistant: Sentiment(Negative)

Human: "My day has been great”:

Assistant: Sentiment(Positive)

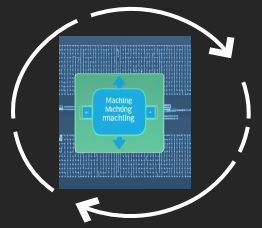
Human: "This is the link to the article”:

Assistant: Sentiment(Neutral)

Human: {{Tweet}}

Assistant: Sentiment(

Advanced Approaches



Chain of Thought Prompting (CoT)

Enable complex reasoning capabilities via intermediate reasoning steps

Include few shot examples to improve results on reasoning tasks

Provide transparency on model outputs

Retrieval Augmented Generation (RAG)

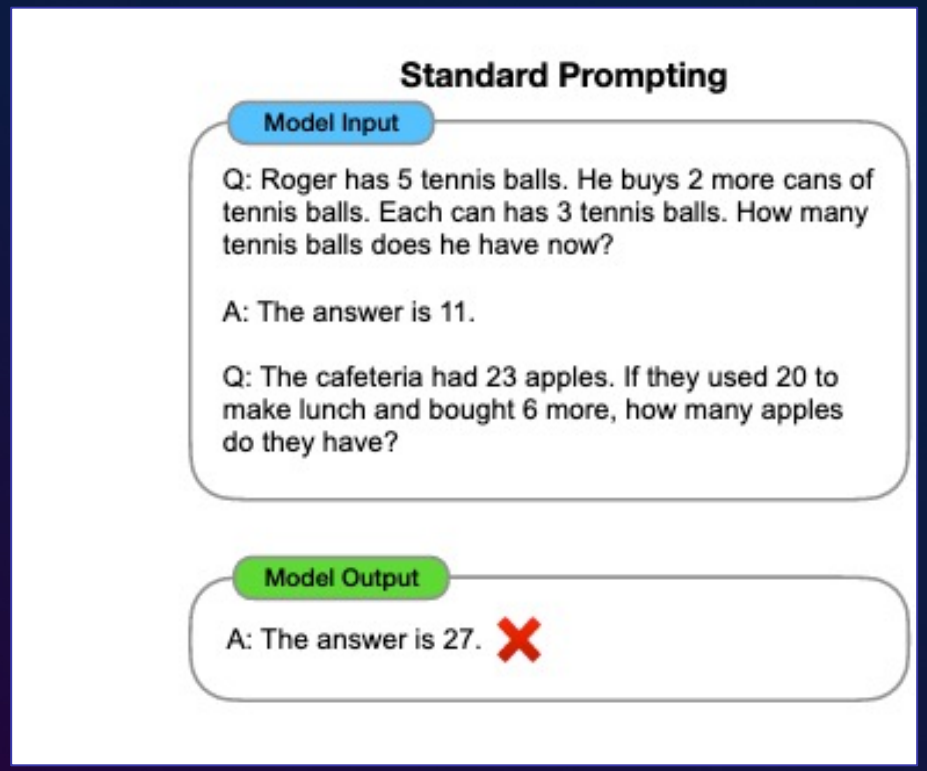
Enrich context of prompts from internal knowledge repositories

Mitigate the effects hallucination using prompt grounding

Best fit for closed domain Q&A and chatbot use cases

Chain of Thought – CoT

- Explore how series of intermediate steps improve ability of LLMs to perform complex reasoning task



1-shot example

5 tennis balls
2 cans, 3 each. 2 x 3 = 6
5 + 6 = 11 tennis balls

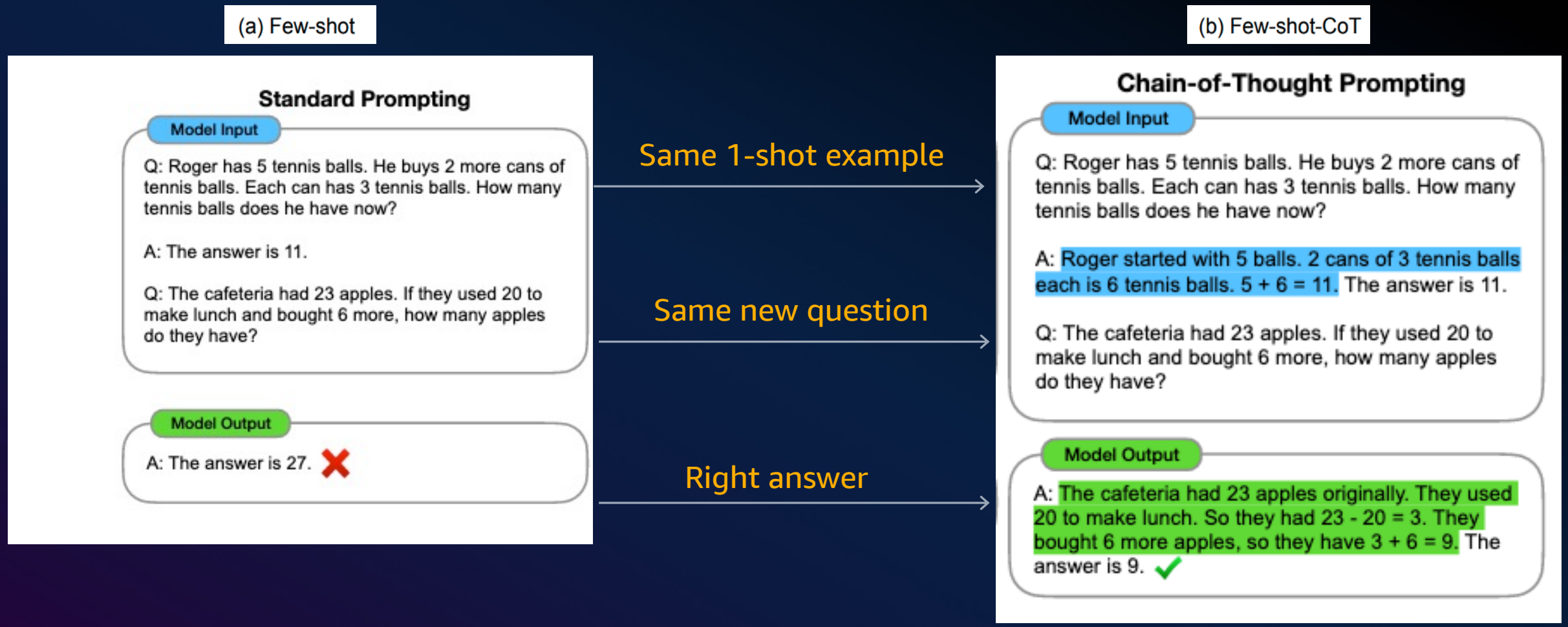
New complex problem

Wrong answer

23 apples
20 used, 3 left
Bought 6
3 + 6 = 9

Source: [Chain-of-Thought Prompting Elicits Reasoning in Large Language Models]

Chain of Thought – CoT (Few-shot-CoT)



Source: [Chain-of-Thought Prompting Elicits Reasoning in Large Language Models]

Chain of Thought – CoT (Zero-shot-CoT)

Giving permission to think:

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: The answer (arabic numerals) is

(Output) 8 ✗

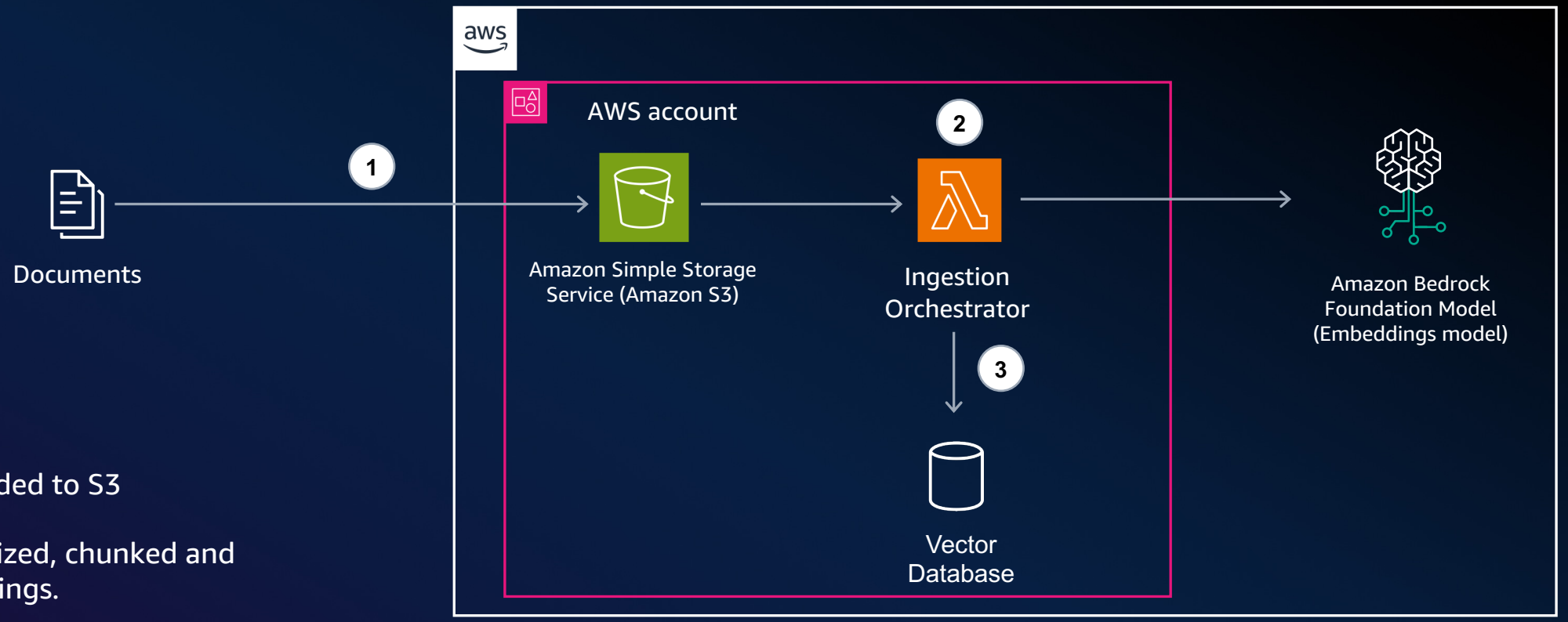
(d) Zero-shot-CoT (Ours)

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A: ***Let's think step by step.***

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

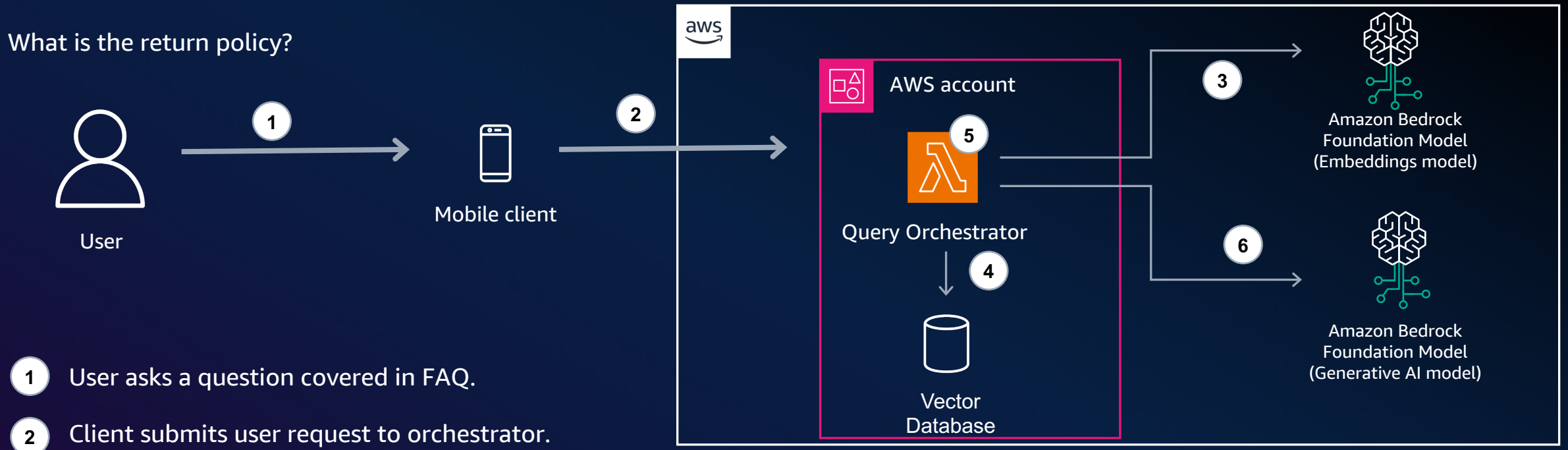
Source: Chain-of-Thought Prompting Elicits Reasoning in Large Language Models

RAG: System Design – Ingestion of Data for Q+A



1. Documents are uploaded to S3
2. Documents are tokenized, chunked and converted to embeddings.
3. Embeddings plus metadata stored into a vector database for retrieval process's similarity search.

RAG: System design – Q+A



1. User asks a question covered in FAQ.
2. Client submits user request to orchestrator.
3. User's question is converted to embeddings.
4. Retrieval - Similarity search performed between customer's question (converted to embeddings) and existing database of FAQs.
5. Augmentation – Search results in original language format brought back to the gen AI model.
6. Generation – Gen AI model will use the results to generate a natural language response.

Prompt engineering guidelines

Parts of a prompt

1. "\n\nHuman:"
2. Task context
3. Tone context
4. Background data to process
5. Detailed task description & rules
6. Examples
7. Immediate data to process
8. Immediate task description or request
9. Thinking step by step / take a deep breath
10. Output formatting
11. "\n\nAssistant:"

Example:

Human: You will be acting as an AI career coach named Joe created by the company AdAstra Careers. Your goal is to give career advice to users. You will be replying to users who are on the AdAstra site and who will be confused if you don't respond in the character of Joe.

You should maintain a friendly customer service tone.

Here is the career guidance document you should reference when answering the user: <guide>{{DOCUMENT}}</guide>

Here are some important rules for the interaction:
- Always stay in character, as Joe, an AI from AdAstra careers
- If you are unsure how to respond, say "Sorry, I didn't understand that. Could you repeat the question?"
- If someone asks something irrelevant, say, "Sorry, I am Joe and I give career advice. Do you have a career question today I can help you with?"

Here is an example of how to respond in a standard interaction:

<example>
User: Hi, how were you created and what do you do?
Joe: Hello! My name is Joe, and I was created by AdAstra Careers to give career advice. What can I help you with today?
</example>

Here is the conversation history (between the user and you) prior to the question. It could be empty if there is no history:
<history> {{HISTORY}} </history>

Here is the user's question: <question> {{QUESTION}} </question>

How do you respond to the user's question?

Think about your answer first before you respond. Put your response in <response></response> tags.

Assistant: [Joe] <response>

Claude-specific prompting

7 key prompt engineering techniques

“Human:” / “Assistant:” formatting

- Claude is trained on **alternating “Human:” / “Assistant:”** dialogue:
  - *Human: [Instructions]*
  - *Assistant: [Claude’s response]*
- For any **API prompt**, you must **start with “\n\nHuman:” and end with “\n\nAssistant:”**

Examples:

Human: Why is the sky blue?

Assistant:

Python

```
prompt = “\n\nHuman: Why are sunsets orange?\n\nAssistant:”
```

Be clear and direct

- Claude responds best to clear and direct instructions
- When in doubt, follow the **Golden Rule of Clear Prompting:** show your prompt to a friend and ask them if they can follow the instructions themselves and produce the exact result you're looking for

Example:

Human: Write a haiku about robots

Assistant: Here is a haiku about robots:

Metal bodies move
Circuits calculate tasks
Machines mimic life

Human: Write a haiku about robots. **Skip the preamble; go straight into the poem.**

Assistant: Metal bodies move
Circuits calculate tasks
Machines mimic life

Use examples

- **Examples are probably the single most effective tool** for getting Claude to behave as desired
- Make sure to **give Claude examples of common edge cases**
- Generally **more examples = more reliable responses** at the cost of latency and tokens

Example:

Human: I will give you some quotes. Please extract the author from the quote block.

Here is an example:
<example>
Quote:
“When the reasoning mind is forced to confront the impossible again and again, it has no choice but to adapt.”
— N.K. Jemisin, The Fifth Season
Author: N.K. Jemisin
</example>

Quote:
“Some humans theorize that intelligent species go extinct before they can expand into outer space. If they're correct, then the hush of the night sky is the silence of the graveyard.”
— Ted Chiang, Exhalation
Author:

Assistant: Ted Chiang

Give Claude time to think

(It works on humans too!)

Human: [rest of prompt] Before answering, please think about the question within <thinking></thinking> XML tags. Then, answer the question within <answer></answer> XML tags.

Assistant: <thinking>

Helps with troubleshooting Claude's logic & where prompt instructions may be unclear

Human: [rest of prompt] Before answering, please think about the question within <thinking></thinking> XML tags. Then, answer the question within <answer></answer> XML tags.

Assistant: <thinking>[...some thoughts]</thinking>

<answer>[some answer]</answer>

Assign roles (aka role prompting)

- Claude sometimes needs **context about what role it should inhabit**
- Assigning roles changes Claude's response in two ways:
  - **Improved accuracy** in certain situations (such as mathematics)
  - **Changed tone and demeanor** to match the specified role

Example:

Human: Solve this logic puzzle. {{Puzzle}}

Assistant: [Gives incorrect response]

Human: **You are a master logic bot designed to answer complex logic problems.** Solve this logic puzzle. {{Puzzle}}

Assistant: [Gives correct response]

Use XML tags to delineate sections

- Disorganized prompts are hard for Claude to comprehend
- Just like section titles and headers help humans better follow information, **using XML tags <></> helps Claude understand the prompt's structure**

Example:

Human: Hey Claude. Show up at 6AM because I say so. Make this email more polite.

Assistant: Dear Claude, I hope this message finds you well…

Human: Hey Claude. **<email>**Show up at 6AM because I say so.**</email>** Make this email more polite.

Assistant: Good morning team, I hope you all had a restful weekend…

Format output & speak for Claude

- You can get Claude to say exactly what you want by:
  - **Specifying the exact output format** you want
  - **Speaking for Claude** Prefill Claude's response by writing text in the "Assistant" field. Claude will continue from where you left off

Example:

Prompt

```
Human: Please write a haiku about {{ANIMAL}}. Use JSON format with the keys as "first_line", "second_line", and "third_line".

Assistant: {
```

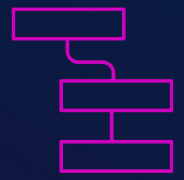
Claude's response

```
  "first_line": "Sleeping in the sun",
  "second_line": "Fluffy fur so warm and soft",
  "third_line": "Lazy cat's day dreams"
}
```

Advanced prompting techniques

Advanced prompting techniques

Chaining prompts | *100K prompts* | *Agents & function calling / tool use* | *Search & RAG*



For tasks with many steps, you can **break the task up and chain together Claude's responses**

Example:

Prompt

Human: Find all the names from the below text:
"Hey, Jesse. It's me, Erin. I'm calling about the party that Joey is throwing tomorrow. Keisha said she would come and I think Mel will be there too."

Assistant <names>

Claude's response

Jesse
Erin
Joey
Keisha
Mel
</names>

a.k.a. {{NAMES}}

Human: Here is a list of names:
<names>{{NAMES}}</names> Please alphabetize the list.

Assistant:

<names>
Erin
Jesse
Joey
Keisha
Mel
</names>

Advanced prompting techniques

Chaining prompts | *100K prompts* | *Agents & function calling / tool use* | *Search & RAG*

For long (10K to 100K token) prompts, do the following:

- When dealing with long documents, **put the doc before the details & query**
- **Put long-form input data in XML tags** so it's clearly separated from the instructions
- Tell Claude to first **find quotes relevant to the question**, then **answer only if it finds relevant quotes**
- Tell Claude to **read the document carefully because it will be asked questions** later
- **Give Claude some example question + answer pairs** that have been generated (either by Claude or manually) **from other parts of the queried text**

Advanced prompting techniques

Chaining prompts | *100K prompts* | *Agents & function calling / tool use* | *Search & RAG*

Function calling, a.k.a. tool use, adds vast functionality to Claude's capabilities by **combining prompts with calls to external functions** that return answers for Claude to use.

Claude **does not** directly call the functions but instead **decides which function to call** and **with what arguments.** The function is then actually called by the client code.

Advanced prompting techniques

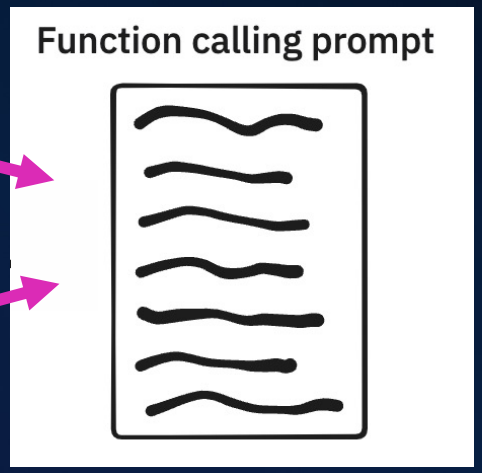
Chaining prompts | *100K prompts* | *Agents & function calling / tool use* | *Search & RAG*

How it works:

Q: What's the weather like in San Francisco right now?

<tool_description>
<tool_name>
get_weather
<tool_name>
</tool_description>

<tool_description>
<tool_name>
play_music
<tool_name>
</tool_description>

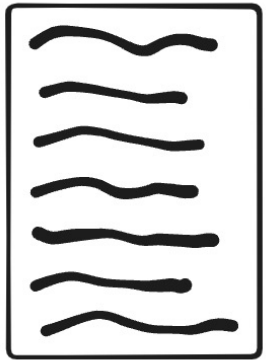


Advanced prompting techniques

Chaining prompts | *100K prompts* | *Agents & function calling / tool use* | *Search & RAG*

How it works cont'd:

Function calling prompt



Claude

Claude judges the relevance of the functions its been given: can it use the functions provided to more accurately answer the question?

YES

```
<function_calls>
<invoke>
<tool_name>get_weather</tool_name>
<parameters>
<location>San Francisco, CA</location>
</parameters>
</invoke>
</function_calls>
```

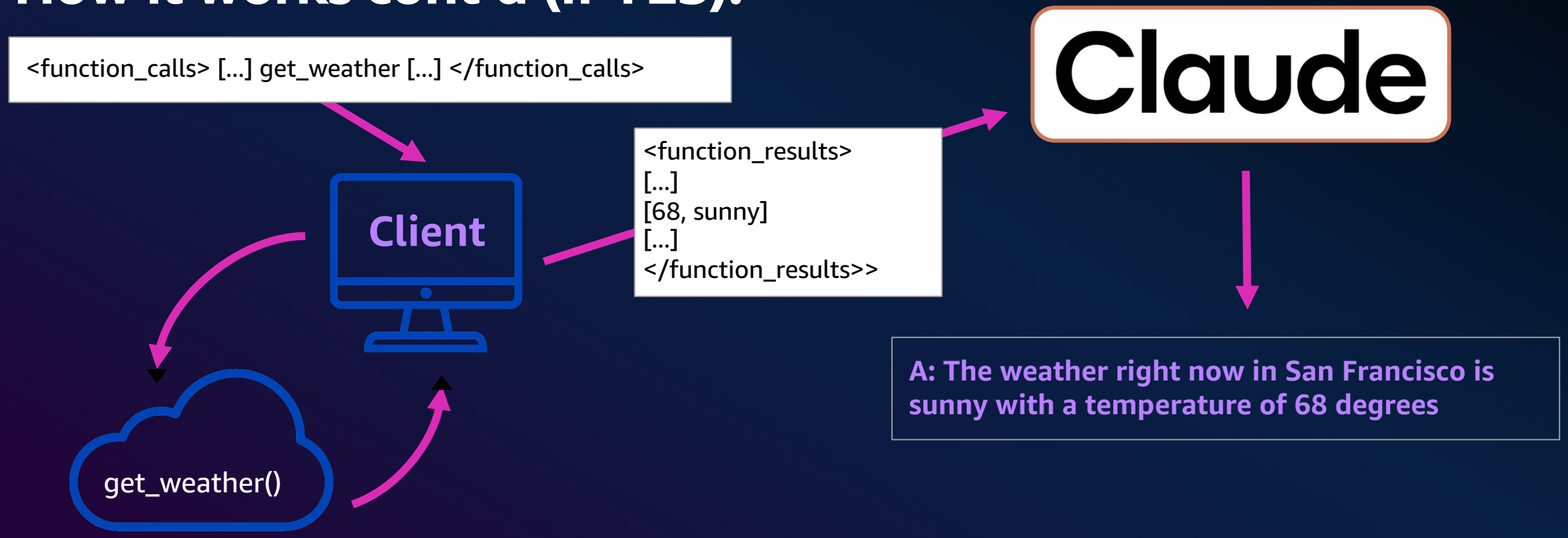
NO

A: I apologize but I don't have access to the current weather in San Francisco.

Advanced prompting techniques

Chaining prompts | *100K prompts* | *Agents & function calling / tool use* | *Search & RAG*

How it works cont'd (if YES):



Advanced prompting techniques

Chaining prompts | *100K prompts* | *Agents & function calling / tool use* | *Search & RAG*

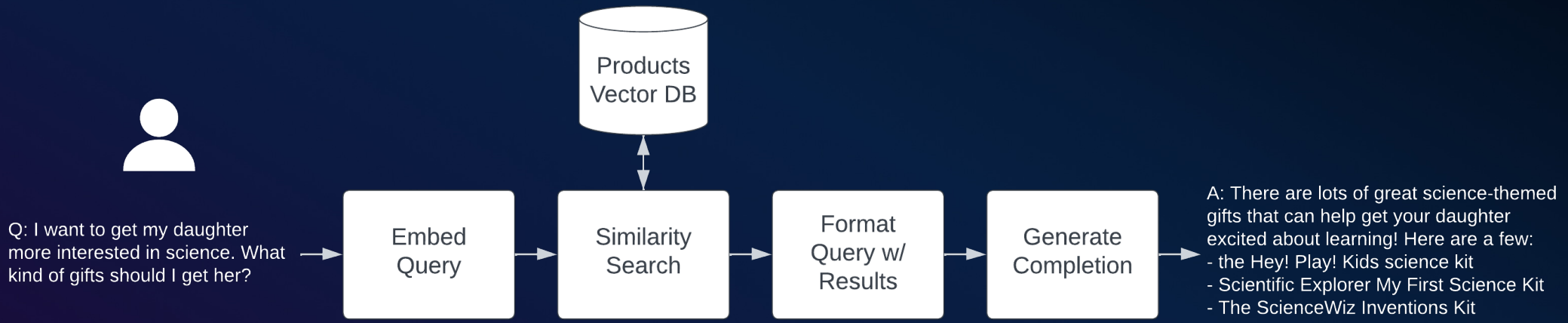
What is retrieval-augmented generation (RAG)?

- Enables the **augmentation of language models with external knowledge**
- **Grounds language model responses in evidence** (i.e., reduces hallucinations)
- **Allows Claude to connect securely to client data**, which increases customizability and analytical precision for client tasks

Advanced prompting techniques

Chaining prompts | *100K prompts* | *Agents & function calling / tool use* | *Search & RAG*

Basic RAG architecture

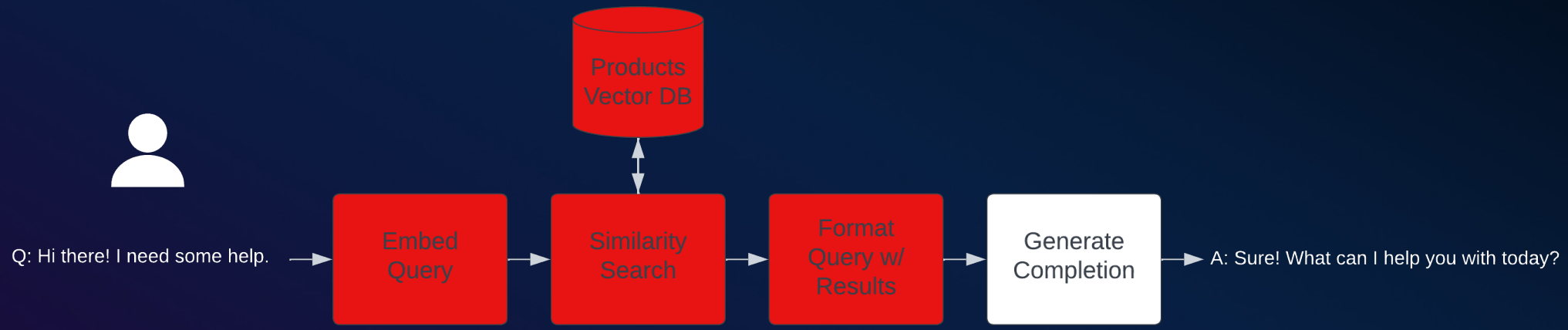


Basic RAG setup. Works great if we know we want to search over a gifts database every time.

Advanced prompting techniques

Chaining prompts | *100K prompts* | *Agents & function calling / tool use* | *Search & RAG*

Traditional RAG w/ Claude

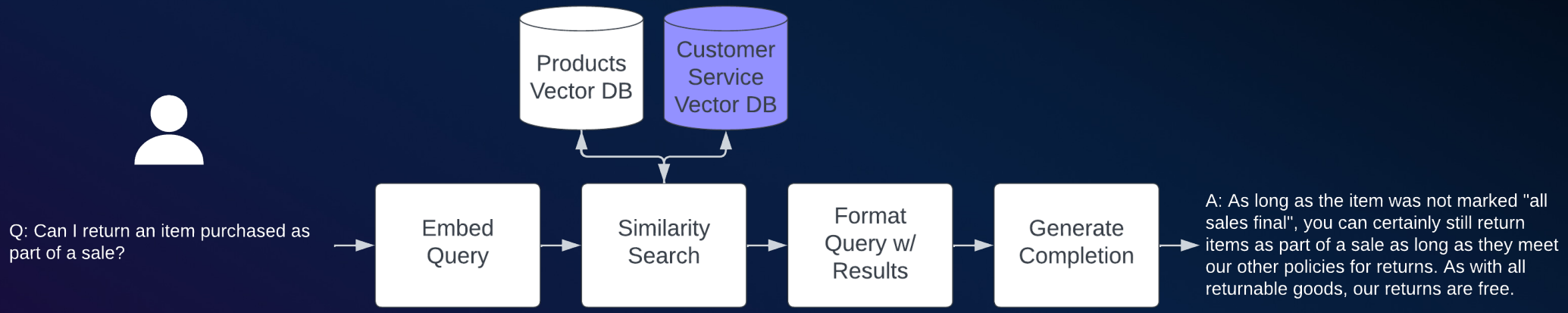


We might want to avoid RAGing if it is not useful in answering the question.

Advanced prompting techniques

Chaining prompts | *100K prompts* | *Agents & function calling / tool use* | *Search & RAG*

Traditional RAG w/ Claude

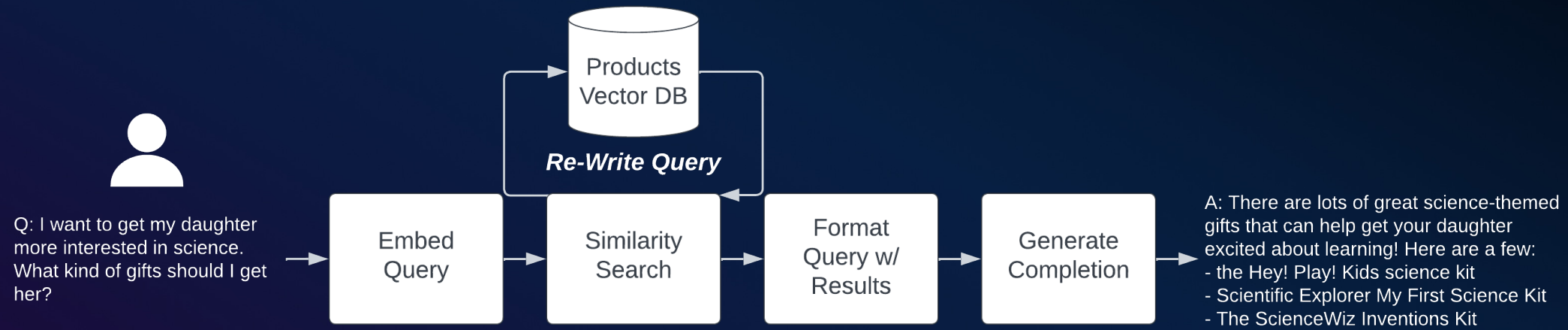


We might want to RAG over different data sources depending on the user query.

Advanced prompting techniques

Chaining prompts | *100K prompts* | *Agents & function calling / tool use* | *Search & RAG*

Traditional RAG w/ Claude

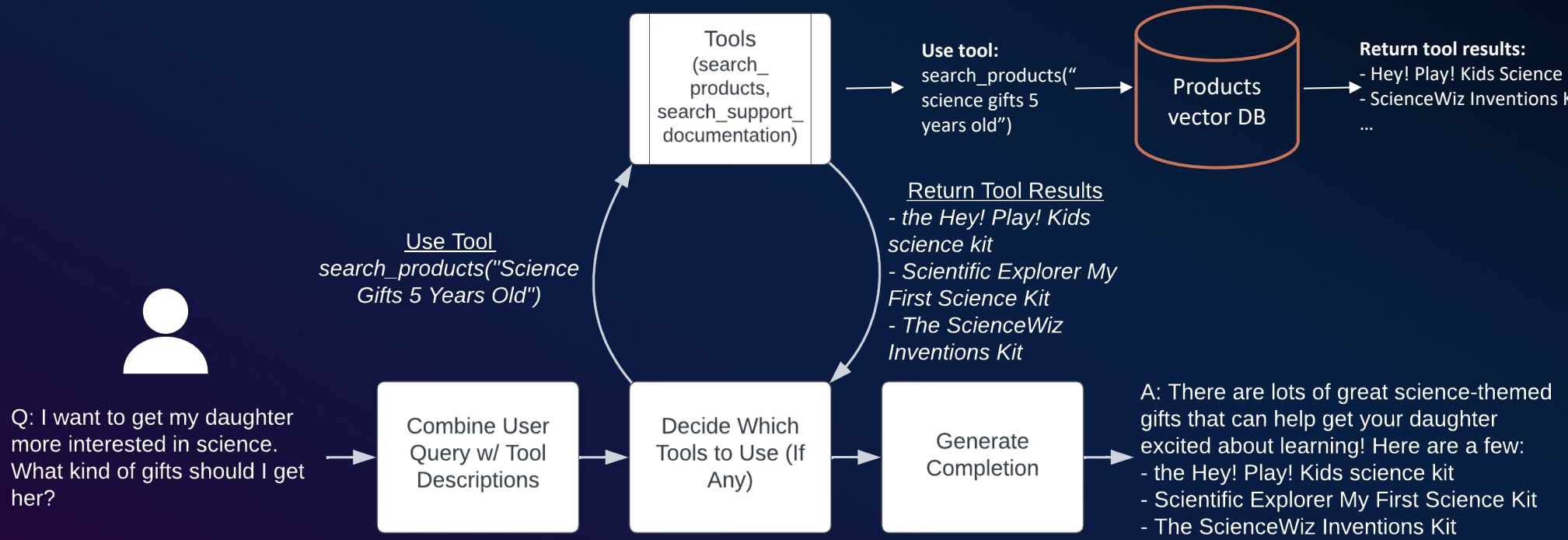


We might want to enable Claude to rewrite its search query and/or requery the data source if it doesn't find what it's looking for the first time.

Advanced prompting techniques

Chaining prompts | *100K prompts* | *Agents & function calling / tool use* | *Search & RAG*

RAG as a tool



skillbuilder.aws

Build beyond

Redeem your free 7-day trial of AWS Skill Builder

Thank you!

Please complete the session survey in the mobile app

Senthilkumar Jeyachandran

AI & ML Specialist Solutions Architect

AWS India