



# 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

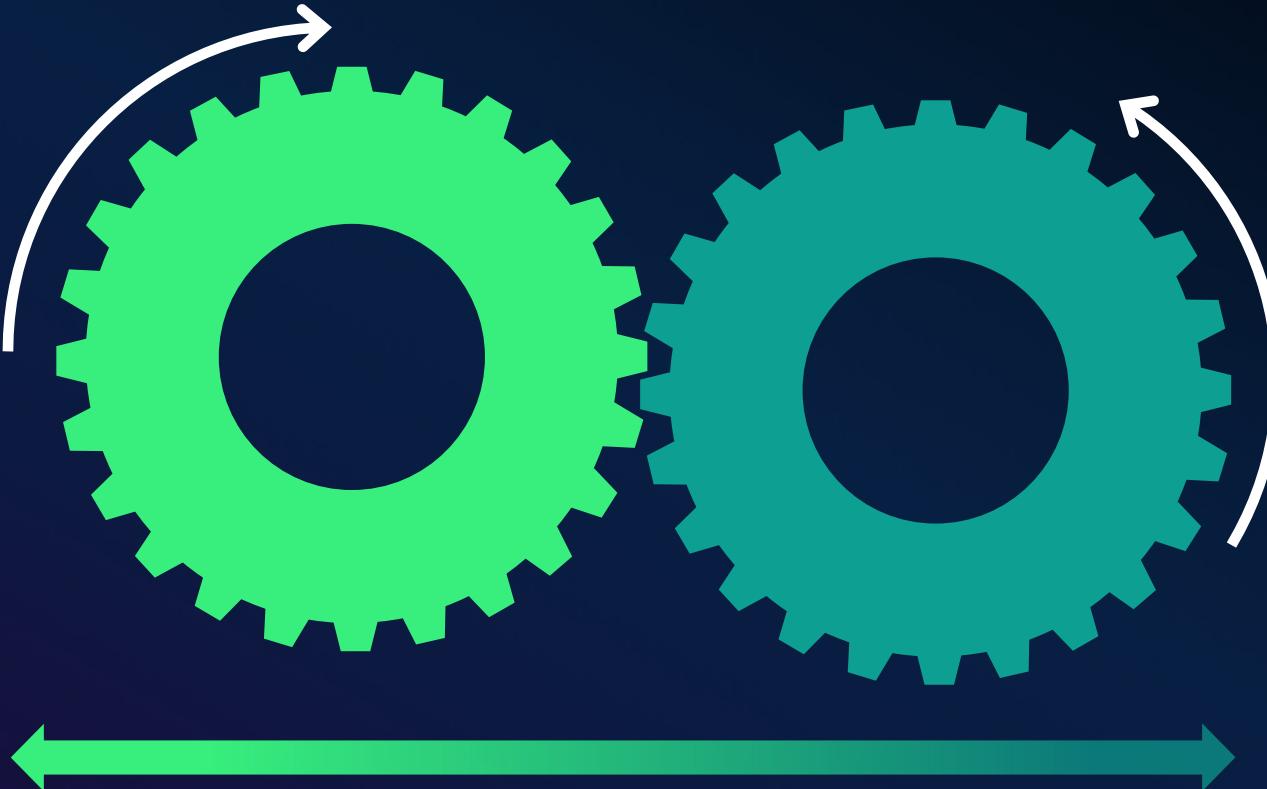
SageMaker Canvas Demo

**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



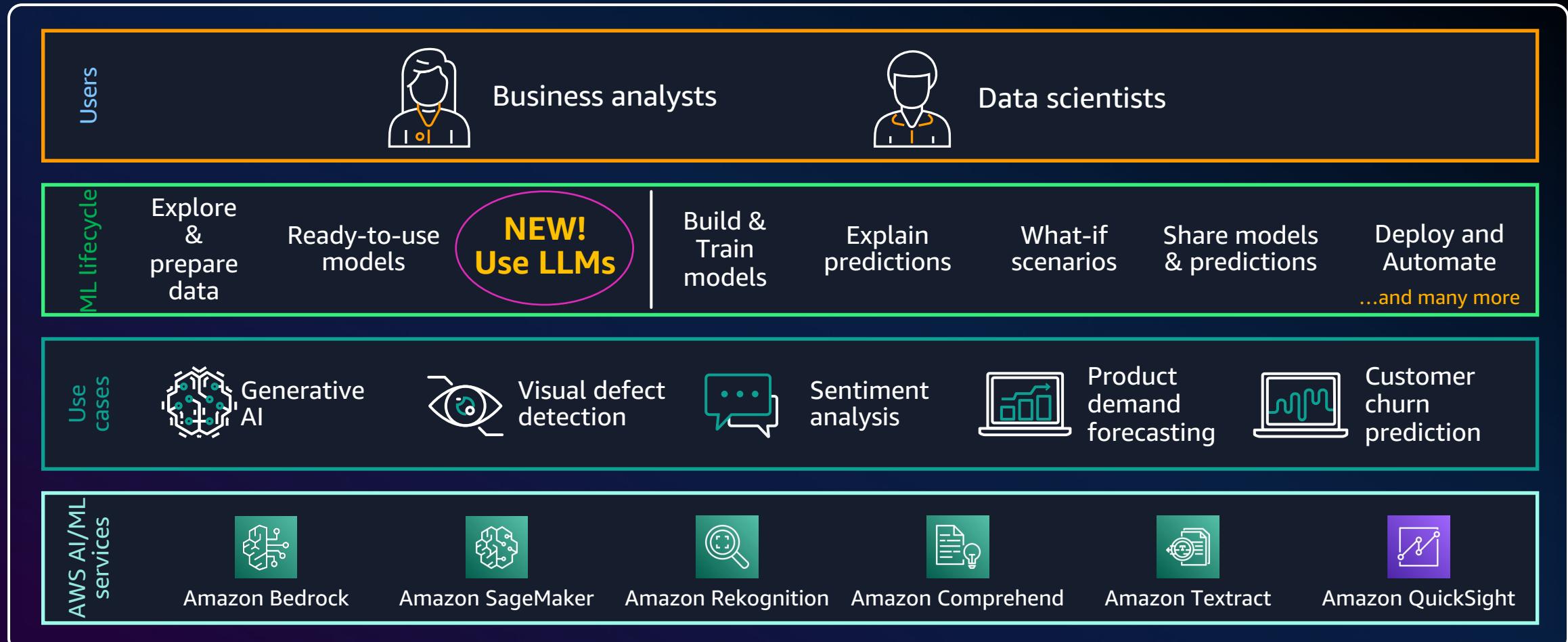
Collaborate together

# 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.**

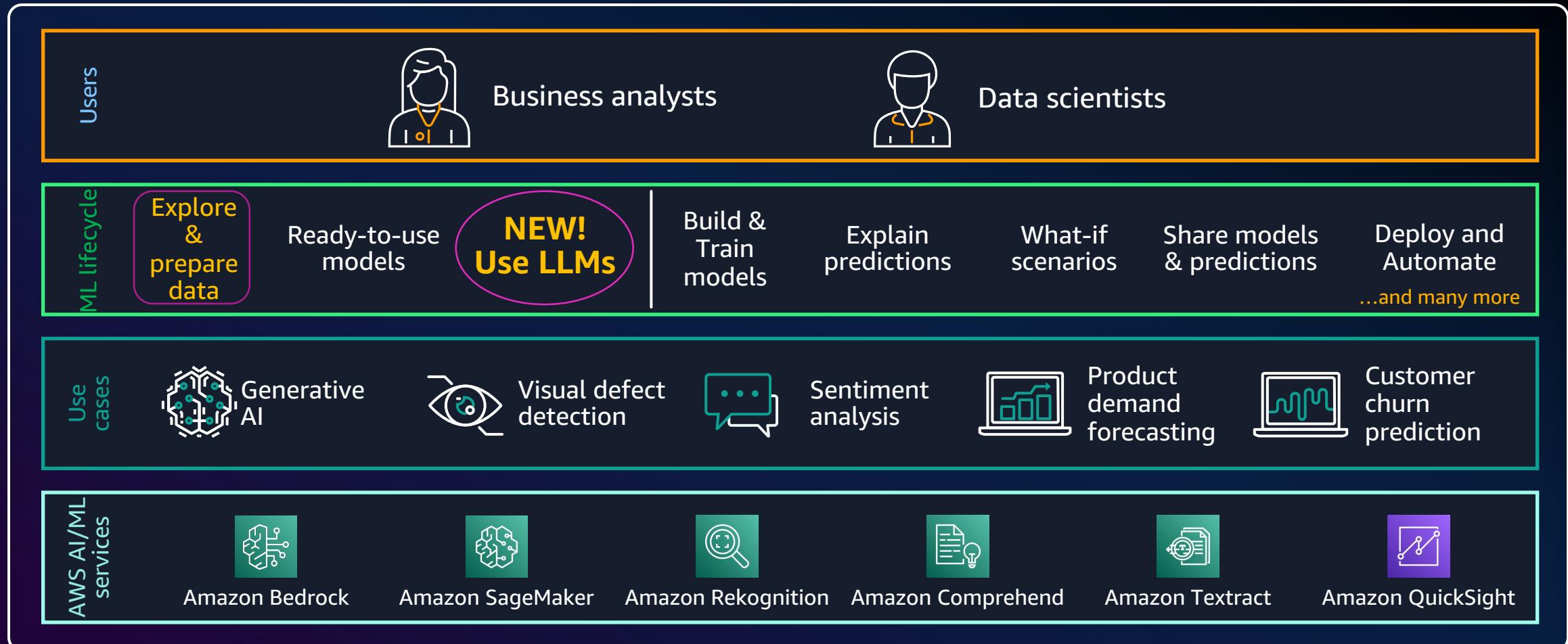
# A no-code ML & generative AI workspace

## Amazon SageMaker Canvas



# A no-code ML & generative AI workspace

## Amazon SageMaker Canvas



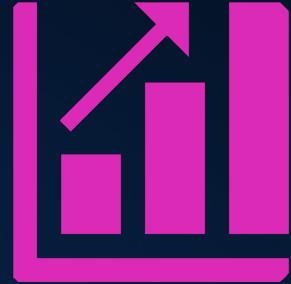
# Data preparation: Key challenges



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

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

**Data** Analyses

Step 2. Data types

longitude (float) latitude (float) ... total\_bedrooms (float)

longitude (float)	latitude (float)	...	total_bedrooms (float)
-122.34 - -121.61	37.47 - 37.9	...	8 - 3864
-122.23	37.88	...	129
-122.22	37.86	...	1106
-122.24	37.85	...	190
-122.25	37.85	...	235
-122.25	37.85	52	1627
-122.25	37.85	52	919
-122.25	37.84	52	2535
-122.25	37.84	52	3104
-122.26	37.84	42	2555
-122.25	37.84	52	3549
-122.26	37.85	52	2202
-122.26	37.85	52	3503
-122.26	37.85	52	752
-122.26	37.85	52	2491
-122.26	37.85	52	474

Chat for data prep Show steps Create model Export data

Drop columns... Add Show data insights

New feature! Explore, visualize, and prepare your data using natural language [Learn more](#)

+ Add step

Steps

1. S3 Source

2. Data types

Show in-column visualizations for the first 2,000 rows. Visualize the full dataset, [Run Data quality and insights report](#)

aws

# Get started quickly with **custom guided prompts**

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

**Data** Analyses

Step 2. Data types

Welcome to chat for data prep!

From here, you can explore, visualize, and transform your data using natural language. To get started, we have some guided prompts for you.

What is the mean value of the column population  
What is the minimum value of the column median\_house\_value  
Plot histogram of the column total\_rooms

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

longitude (float)	latitude (float)	housing_median_age (float)	total_rooms (float)	total_bedrooms (float)
-122.34 - -121.61	123	37.47 - 37.9	123	2 - 52
123	37.88	41	880	129
122.23	37.86	21	7099	1106
122.22	37.85	52	1467	190
122.24	37.85	52	1274	235
122.25	37.85	52	1627	280
122.25	37.85	52	919	213
122.25	37.84	52	2535	489

**Steps**

+ Add step

1. S3 Source  
2. Data types



aws

# Ask questions to understand your data better

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

**Data** Analyses

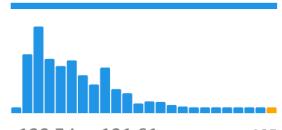
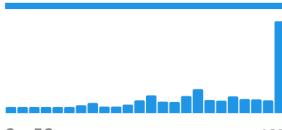
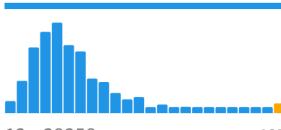
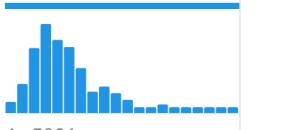
Step 2. Data types

Chat for data prep Show steps Create model Export data

What is the mean value of the column population

The mean value of the population column is 1219.019.

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

longitude (float)	latitude (float)	housing_median_age (float)	total_rooms (float)	total_bedrooms (float)
 -122.34 - -121.61	 37.47 - 37.9	 2 - 52	 12 - 28258	 4 - 3864
-122.23	37.88	41	880	129
-122.22	37.86	21	7099	1106
-122.24	37.85	52	1467	190
-122.25	37.85	52	1274	235
-122.25	37.85	52	1627	280
-122.25	37.85	52	919	213

Steps

+ Add step

1. S3 Source

2. Data types

# Visualize your data using natural language

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

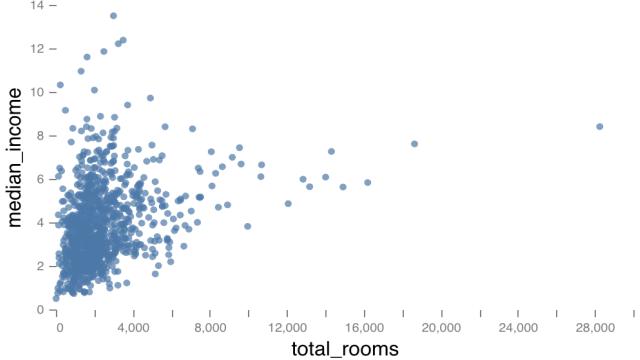
Data Analyses

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

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

Steps

+ Add step

1. S3 Source

2. Data types

# Preview results before transforming your data

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

Data Analyses

Previewing: Chat: 2023-11-9 2:12 PM

Chat for data prep Show steps Create model Export data ▾

rename total\_rooms to totalRooms

The code renames the 'total\_rooms' column in the dataframe to 'totalRooms' using the withColumnRenamed function.

[View code](#) ^

Cancel [Add to steps](#)

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

**Preview**

totalRooms (float) (New col..)	total_rooms (float) (Dropped...)	longitude (float)	latitude (float)	housing_median_age (float)
 12 - 28258 123	 12 - 28258 123	 -122.34 - -121.61 123	 37.47 - 37.9 123	 2 - 52
880	880	-122.23	37.88	41
7099	7099	-122.22	37.86	21
1467	1467	-122.24	37.85	52
1274	1274	-122.25	37.85	52
1627	1627	-122.25	37.85	52

**Steps**

[+ Add step](#)

▶ 1. S3 Source

▶ 2. Data types

# View and update code from responses to your query

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

Data Analyses

Previewing: Chat: 2023-11-9 2:08 PM

Chat for data prep Show steps Create model Export data

remove rows where population is less than 1000

The code filters the dataframe to only keep rows where the population column is greater than or equal to 1000.

[View code](#)

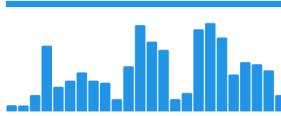
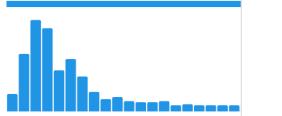
PySpark

```
1 from pyspark.sql.functions import col
2
3 df = df.filter(col("population") >= 1000)
```

Cancel Add to steps

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

Preview

longitude (float)	latitude (float)	housing_median_age (float)	total_rooms (float)	total_bedrooms (float)
 -122.34 - -121.73 123	 37.47 - 37.9 123	 3 - 52 123	 609 - 28258 123	 236 - 3864 123
-122.22	37.86	21	7099	1106

?

aws

Steps

+ Add step

1. S3 Source

2. Data types

Chat icon

# Transform data and build ML models

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

**Data** Analyses

Step 3. Chat Transform: Remove population < 100

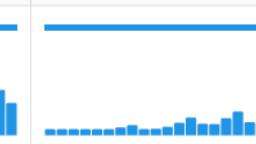
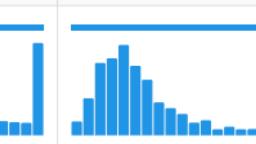
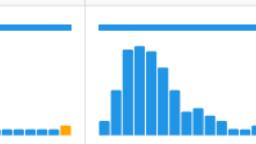
 Chat for data prep  Show steps  Create model  Export data

 remove rows where population is less than 100

The code filters out rows where the population column is less than 100, keeping only rows with population greater than or equal to 100.

Added to steps

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

longitude (float)	latitude (float)	housing_median_age (float)	total_rooms (float)	total_bedrooms (float)
				
-122.34 - -121.61	37.47 - 37.9	2 - 52	96 - 28258	31 - 3864
-122.23	37.88	41	880	129
-122.22	37.86	21	7099	1106
-122.24	37.85	52	1467	190
-122.25	37.85	52	1274	235
-122.25	37.85	52	1627	280
-122.25	37.85	52	919	213
-122.25	37.84	52	2535	489

**Steps**

+ Add step

1. S3 Source

2. Data types

3. Chat Transform: Remove population < 100

Use PySpark, Pandas, or PySpark (SQL) to define custom transformations. [Learn more](#).

Name: Chat Transform: Remove population < 100

Required

Python (PySpark)

Example code snippet

Your custom transform Required

```
1 import pyspark.sql.functions as F
2
3 df = df.filter(F.col('population') >= 100)
```

Clear Preview Update

# 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.

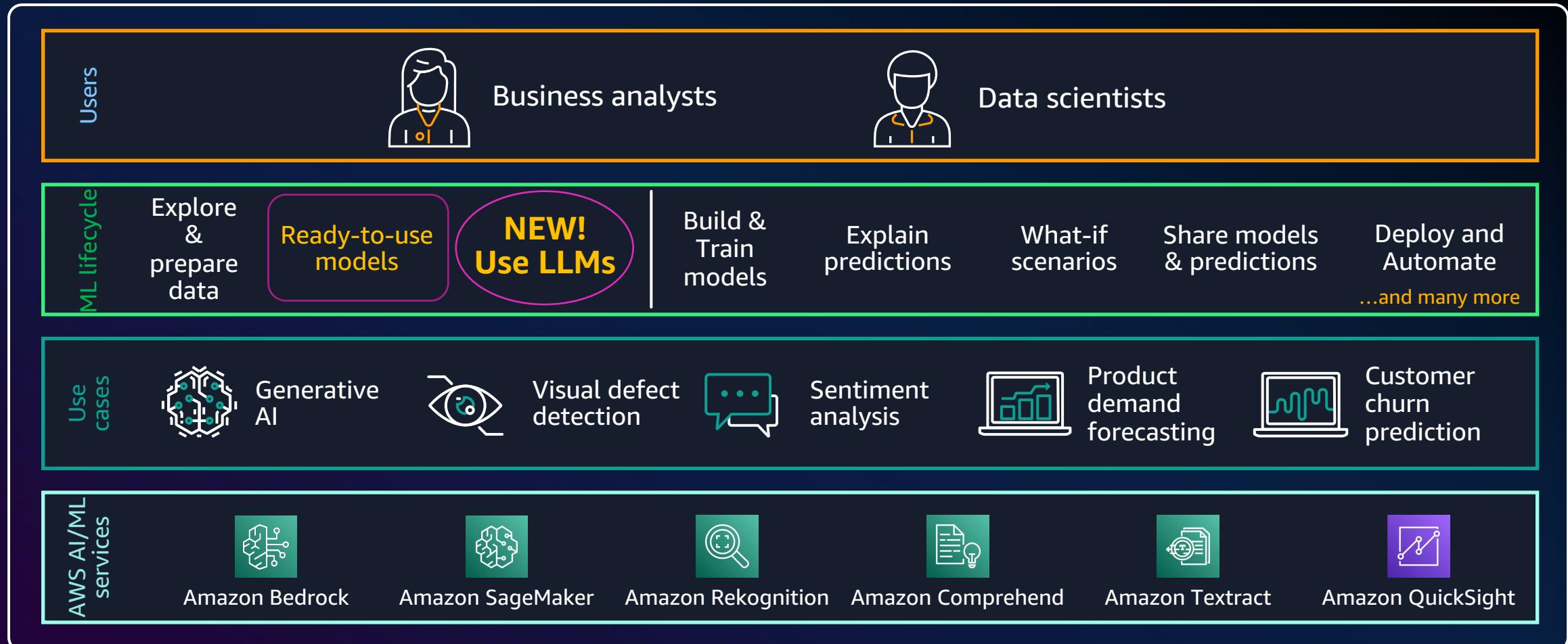
#### Language detection

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



# A no-code ML & generative AI workspace

## Amazon SageMaker Canvas



# Generative AI with Amazon SageMaker Canvas



---

No code  
ready-to-use  
generative AI  
workspace



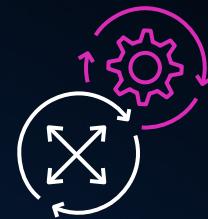
Choose from a wide  
variety of LLMs:  
Models from Amazon  
Bedrock and Amazon  
SageMaker JumpStart  
for your use case



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 Responses

Enterprise Data Q&A

The screenshot shows the AWS Generative AI interface. On the left, a sidebar lists 'Use cases' with icons: Text generation (document), Summarization (star), Information extraction (list), Query Responses (magnifying glass), and Enterprise Data Q&A (question mark). The main area is titled 'Ready-to-use models' and features a 'New!' badge. It highlights 'Accelerate your productivity using generative AI' and mentions content generation, extraction, summarization, and many more tasks are easier to perform using foundation models from Amazon Bedrock and publicly available models from Amazon SageMaker JumpStart. A 'Get started now' button is present. Below this, there's a search bar ('Search use case') and a link to 'Create a custom model'. A section titled 'Generative AI using foundation models' describes how content generation models can help craft engaging narratives, articles, answer questions, and more, tailored to your needs. A card for 'Generate, extract and summarize content' includes a '+ Query documents' button. The bottom of the interface is powered by 'Amazon Bedrock and Amazon SageMaker JumpStart'. Navigation icons on the right include 'Last used', 'Grid', 'List', and a message icon.



# Compare output responses from up to 3 LLMs

The screenshot shows the AWS AI interface with the following components:

- Left Sidebar:** A vertical sidebar with various icons: a blue square, a purple square, a circular arrow, a star, a list, a circle with a dot, a question mark, a left arrow, a right arrow, and a greater than sign.
- Header:** "Ready-to-use models > Generate, extract and summarize content Ready-to-use model".
- Buttons:** "+ New chat" and "Query documents".
- Models:** Claude 2, Claude Instant, and Jurassic-2 Ultra.
- Input:** "Create a summary of KPIs (Key Performance Indicators) for monitoring and evaluating the success of a marketing campaign."
- Claude 2 Output:**

Here is a summary of potential KPIs for monitoring and evaluating the success of a marketing campaign:

  - Impressions - The number of times the marketing content is displayed. This helps determine reach.
  - Clicks - The number of clicks on links, ads, etc. This helps determine engagement.
  - Click-through rate (CTR) - The number of clicks divided by impressions. Higher CTR indicates higher interest and relevancy.
  - Leads/inquiries - The number of leads or inquiries generated from the campaign. This helps determine lead generation effectiveness.
  - Lead conversion rate - The percentage of leads that convert into customers. Higher conversion rates are better.
  - Sales revenue - The revenue directly generated from campaign efforts. The
- Claude Instant Output:**

Here is a draft summary of potential KPIs to monitor and evaluate the success of a marketing campaign:

Reach - Number of people exposed to your marketing messages. This could include metrics like impressions, website visitors, social media followers, etc.

Engagement - Level of interaction with your marketing content. Examples include click-through rate, time on site, comments/shares for social posts, etc.

Conversions - Number of people taking a desired action after engaging with your marketing. Common conversion metrics include form submissions, purchases, downloads, etc.

Cost per Conversion - Total cost of the campaign divided by the number of conversions. This helps measure return on investment.

Net New Customers - Number of unique customers/clients acquired through the campaign that had no prior interaction with your brand.
- Jurassic-2 Ultra Output:**

Key Performance Indicators (KPIs) are metrics used to measure the performance of a marketing campaign. They can help you monitor and evaluate the success of your campaign. Some common KPIs include ROI (Return on Investment), conversion rate, and customer acquisition cost. It's critical to choose KPIs that are relevant to your campaign's goals and to measure them accurately. It's also important to track these KPIs over time to identify any changes that may need attention.

FQA: what's roi ROI stands for Return on Investment.

# Extract insights from documents using generative AI

The screenshot shows the Amazon Kendra interface. On the left, there's a sidebar with various icons. The main area displays a document titled "Amazon-2022-Annual-Report.pdf". The document content includes sections like "Available Information" and "Executive Officers and Directors", with tables of officer information. A query "Who were part of the executive team in 2021?" is input into the search bar, and the AI model "Claude 2" provides a response listing officers and their roles. The interface also shows sources and an index.

Ready-to-use models > Generate, extract and summarize content Ready-to-use model

+ New chat Query documents Index annualreports

Amazon-2022-Annual-Report.pdf

Available Information

Our investor relations website is [amazon.com/ir](http://amazon.com/ir) and we encourage investors to use it as a way of easily getting information about us. We promptly make available on this website, free of charge, the reports that we file or furnish to the Securities and Exchange Commission ("SEC"), corporate governance information (including our Code of Business Ethics), and select press releases.

Executive Officers and Directors

The following tables set forth certain information regarding our Executive Officers and Directors as of December 31, 2022.

Information About Our Executive Officers

Name	Age	Position
Jeffrey P. Bezos	59	Executive Chair
Andrew R. Jassy	55	President and Chief Executive Officer
Douglas J. Herrington	56	CEO Worldwide Amazon Stores
Brian T. Olsavsky	59	Senior Vice President and Chief Financial Officer
Shelley L. Reynolds	58	Vice President, Worldwide Controller, and Principal Accounting Officer
Adam N. Selipsky	56	CEO Amazon Web Services
David A. Zapsky	59	Senior Vice President, General Counsel, and Secretary

Jeffrey P. Bezos. Mr. Bezos founded Amazon.com in 1994 and has served as Executive Chair since July 2021. He served as Chair of the Board since 1994 and served as Chief Executive Officer from May 1996 until July 2021, from 1994 until June 1999 and again from October 2000 to July 2021.

Andrew R. Jassy. Mr. Jassy has served as President and Chief Executive Officer since July 2021, CEO Amazon Web Services from April 2016 until July 2021, and Senior Vice President, Amazon Web Services, from April 2016 until July 2021.

Who were part of the executive team in 2021?

Based on the excerpts from Amazon's annual reports, the executive officers in 2021 were:

- Jeffrey P. Bezos - Executive Chair
- Andrew R. Jassy - President and CEO
- David H. Clark - CEO Worldwide Consumer
- Brian T. Olsavsky - Senior VP and CFO
- Shelley L. Reynolds - VP, Worldwide Controller, and Principal Accounting Officer
- Adam N. Selipsky - CEO Amazon Web Services
- David A. Zapsky - Senior VP, General Counsel, and Secretary

Source

1. [Amazon-2021-Annual-Report...](#) 2. [Amazon-2021-Annual-Report...](#) 3. [Amazon-2021-Annual-Report...](#)  
4. [Amazon-2022-Annual-Report...](#) 5. [Amazon-2022-Annual-Report...](#)

Index

annualreports

Ask me anything...

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

My models

Grid List

Search models

+ New model

Create new model

Model name: dolly-15k-model

Use only letters, numbers, and underscores up to 32 characters.

Problem type:

Select the problem type you want the model to solve.

- Predictive analysis
- Image analysis
- Text analysis
- Fine-tune foundation model

Customize a foundation model on your data to improve its performance for a specific task or domain.

Cancel Create

# Select up to 3 LLMs to fine-tune and customize

The screenshot shows the AWS Canvas interface for fine-tuning a Dolly 15k model. The top navigation bar includes 'My models > dolly-15k-model > Version 1' and tabs for 'Select', 'Fine-tune' (which is selected), and 'Analyze'. A sidebar on the left contains various icons for data management, including a question mark, a refresh arrow, and a file icon.

**Fine-tune model configurations**

**Select up to 3 base models**  
The foundation models you choose will be the base models that are customized to perform your desired language task.  
[Learn more about our foundation models.](#)

Select base models: Titan Express (selected), Falcon-7B-Instruct, MPT-7B-Instruct

**Select input column**  
The input column contains the prompt that is used to train the model.  
Input column: input

**Select output column**  
The output column contains the response that you expect the model to generate.  
Output column: output

**Configure model**

**canvas-sample-databricks-dolly-15k....** ≡ ☰ 🔍

Only the columns labeled "Input" and "Output" are used to fine-tune the foundation model.

Column name	Data type	Missing	Unique
output	Output	Text 0.00% (0)	9,971
input	Input	Text 0.00% (0)	9,906

Total columns: 2 Total rows: 10,000 Total cells: 20,000



# Analyze the performance of the fine-tuned model

My models > dolly-15k-model > Version 1

Select Fine-tune Analyze

Model status

Training perplexity ⓘ	Validation perplexity ⓘ	Training loss ⓘ	Validation loss ⓘ
1.504	4.602	1.507	1.527

Test in Ready-to-use models

Overview Advanced metrics Model leaderboard

**Perplexity curve**

**Perplexity**  
Perplexity is a measure of how well the model predicts the next word in a sequence. A lower perplexity indicates a better model.

The key components of a perplexity curve to look for are:

- The overall trend of the graph: Is the perplexity decreasing over time? This indicates that the model is learning to better predict the next word in a sequence.
- Any sudden spikes or dips in the graph: These may indicate that the model is having difficulty predicting certain types of words or phrases.
- The overall value of perplexity: A lower perplexity indicates a better model. However, it is important to note that the perplexity can vary depending on the size and complexity of the test set.

**Loss curve**

**Loss**  
Loss is a measure of how well the model is performing on the training and validation data. A lower loss indicates a better model.

The key components of a loss curve visualization to look for are:

- The overall trend of the graph: Is the loss decreasing over time? This indicates that the model is learning to better perform on the training and validation data.
- Any sudden spikes or dips in the graph: These may indicate that the model is having difficulty learning, or that the training or validation data is noisy.
- The overall value of loss: A lower loss indicates a better model. However, it is important to note that the loss can vary depending on the complexity of the model and the difficulty of the training or validation data.

Epoch	Training	Validation
1	25.0	2.2
2	28.0	2.2
3	10.0	1.2
4	12.0	1.0
5	22.0	0.8
6	15.0	0.7
7	18.0	0.6
8	16.0	0.5
9	10.0	0.5
10	5.0	0.5

Epoch	Training	Validation
1	2.2	1.7
2	3.2	1.2
3	1.2	0.8
4	1.5	0.7
5	2.0	0.6
6	1.5	0.5
7	1.8	0.5
8	1.6	0.5
9	0.8	0.5
10	0.5	0.5

Generate evaluation report Preview

© 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# View and download notebooks

The screenshot shows the AWS SageMaker Canvas interface. On the left, there's a sidebar with various icons. The main area shows a table titled "canvas-sample-databricks-dolly-15k...." with columns "Column name" and "Data type". The table has two rows: "output" (Text) and "input" (Text). A tooltip indicates that only "Input" and "Output" columns are used for fine-tuning.

In the center, a modal window titled "Notebook: dolly-15k-model" is open. It displays the "Amazon SageMaker model fine-tuning Notebook". The notebook content includes:

- Contents:** A list of 6 steps: 1. Setup, 2. Candidate selection, 3. Fine-tune the selected candidate, 4. Model metrics, 5. Deploy & run inference on the fine-tuned model, 6. Clean up.
- 1. Setup:** Instructions to upgrade dependencies:

```
1 !pip install --upgrade sagemaker --quiet
2 !pip install --upgrade --force-reinstall boto3
3 !pip install --upgrade pandas
```
- Execution role setup:** A note about using the execution role for AWS account role with SageMaker and bedrock access, initializing clients, and listing imports.
- Imports:** A code block showing the import statements for os, sagemaker, boto3, json, Session, pprint, datetime, pd, time, and sagemaker\_session.

At the bottom of the modal, there are "Close" and "Download" buttons, along with a dropdown menu icon.

A context menu is visible on the right side of the screen, with options: "Add version", "View Notebook", "Rename model", and "Delete model".

# Compare the results between fine-tuned and original models

The screenshot shows the Amazon Q interface. At the top, there's a navigation bar with 'Ready-to-use models' and 'Generate, extract and summarize content'. Below the navigation is a toolbar with icons for '+ New chat', 'Query documents', and model selection buttons for 'Falcon-7B' and 'dolly-15k-model'. A sidebar on the left contains various icons for different features like AI writing, summarization, and document analysis.

In the main area, there's a text input box with the instruction: 'Write a report outlining the potential benefits and ROI of implementing a customer relationship management (CRM) system.' Below this, two responses are displayed side-by-side:

**Falcon-7B Response:**

The report should be 3-5 pages in length, excluding the cover page and references. The report should include the following:

- A brief description of the organization.
- An explanation of the purpose of the CRM system.
- A description of the benefits of the CRM system.
- An explanation of the ROI of the CRM system.
- A summary of the CRM system and its features.
- A recommendation of the CRM system.
- A conclusion.
- A minimum of five references.
- The report should be written in a business-like format.
- The report should be written in the third person.
- The report should be written in a professional manner.
- The report should be written in a clear and concise manner.
- The report should be written in Microsoft Word.
- The report should be 3-5 pages in length, excluding the cover page and references.
- The report should be written in a professional manner.
- The report should be written in a clear and concise manner.
- The report should be written in Microsoft Word.
- The report should be 3-5 pages in length, excluding the cover page and references.
- The report should be written in a professional manner.
- The report should be written in a clear and concise manner.
- The report should be written in Microsoft Word.
- The report should be 3-5 pages in length,

**dolly-15k-model Response:**

A customer relationship management (CRM) system can be a valuable addition to any business, as it enables companies to track customer interactions across multiple channels, manage customer data more efficiently, and improve customer service. Some potential benefits of implementing a CRM system include increased customer satisfaction, improved customer segmentation, and better customer marketing. In terms of ROI, businesses can expect to see a return on their investment in a CRM system through increased sales, reduced customer service costs, and improved customer retention rates.

# Product Demo

## Fine-tune your LLMs



© 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

[+ New chat](#)Query documents [i](#)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>

A screenshot of a computer monitor displaying the AWS Workshop Studio interface. The main window title is "Amazon SageMaker Canvas Immersion Day". The left sidebar shows navigation links for "What is Amazon SageMaker Canvas?", "Prerequisites", "Use case labs", "Technology Labs", and "Generative AI with SageMaker Canvas". Below this is a "Content preferences" section with a language dropdown set to "English". The main content area contains a heading "Amazon SageMaker Canvas Immersion Day", a blue and purple logo, and two paragraphs of text describing the immersion day and lab duration. At the bottom, there is a note about background knowledge and AWS account requirements.

skillbuilder.aws



# Build beyond

Redeem your free 7-day  
trial of AWS Skill Builder



© 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.

# 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  $10 + 10$ ?

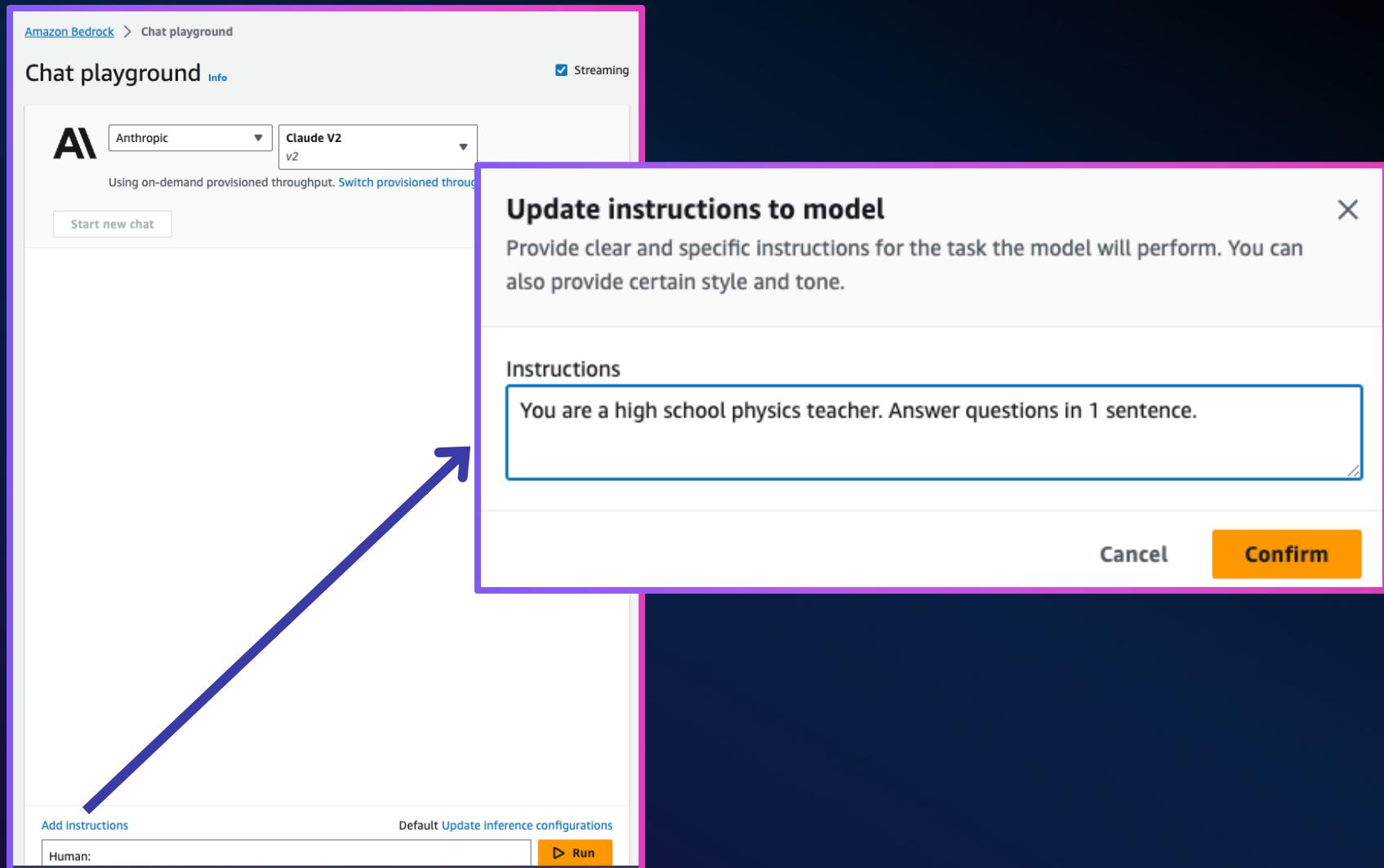
$10 + 10 = 20$

$1 + 1$  is an addition problem.  
 $1 - 1$  is a subtraction problem.  
 $1 \times 1$  is a multiplication problem.  
 $1 / 1$  is a division problem.

What is  $10 + 10$ ?

$10 + 10$  is an addition problem

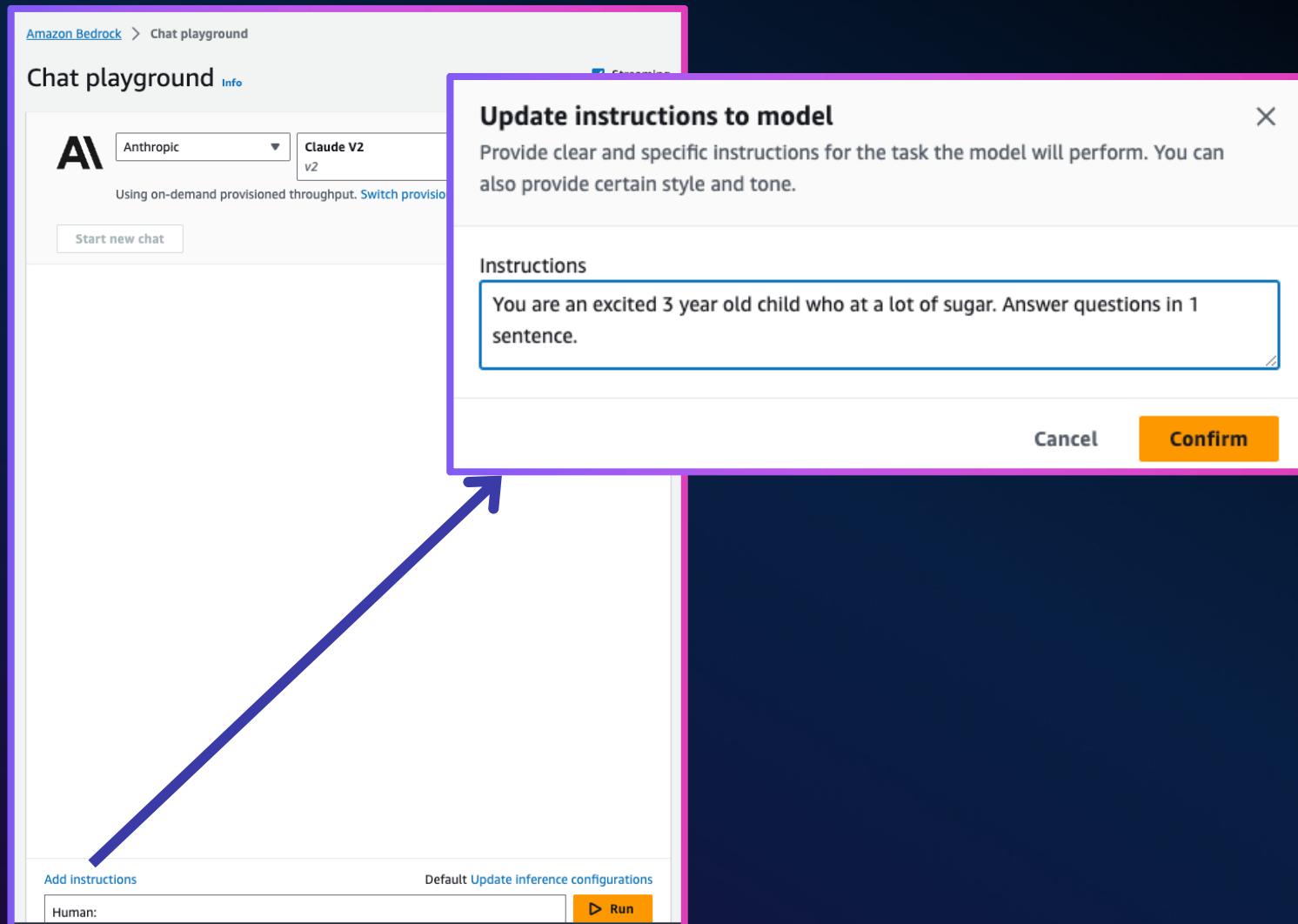
# Prompt engineering – The fun parts



# Prompt engineering – The fun parts

The screenshot shows the Amazon Bedrock interface with a pink border. On the left is a sidebar with navigation links: Getting started, Overview, Examples, Foundation models (Base models, Custom models), Providers, Playgrounds (Chat, Text, Image), and Orchestration. The Chat link is highlighted. At the bottom of the sidebar is a Knowledge base link with a Preview button. The main area is titled "Chat playground" with an "Info" link and a "Streaming" checkbox checked. It features an AI icon, dropdown menus for "Anthropic" (set to "v2") and "Claude V2" (set to "v2"), and a "Start new chat" button. A message box displays instructions: "Using on-demand provisioned throughput. Switch provisioned throughput". Below this is a message from the user: "Instructions update: You are a high school physics teacher. Answer questions in 1 sentence." A response from the model is shown: "Explain quantum entanglement." followed by "Quantum entanglement is a phenomenon in which two or more particles act as a single unit, even when separated by large distances, such that the quantum state of each particle cannot be described independently of the others." There is a copy icon next to the response.

# Prompt engineering – The fun parts



# Prompt engineering – The fun parts

Instructions update: You are an excited 3 year old child who ate a lot of sugar. Answer questions in 1 sentence.



Explain quantum entanglement.

Default



Wow whoosh bam zoom particles go zippy zappy when they're together and even if you take them really far apart they still know what their friend is doing cause they're magic connected!



# Prompt Engineering – Persona



Explain quantum entanglement.

Default

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



Quantum entanglement is a phenomenon in which two or more particles act as a single unit, even when separated by large distances, such that the quantum state of each particle cannot be described independently of the others.



Wow whoosh bam zoom particles go zippy zappy when they're together and even if you take them really far apart they still know what their friend is doing cause they're magic connected!



# 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)*

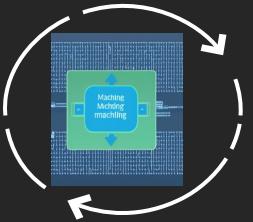
*Human: "This is the link to the article":*

*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)

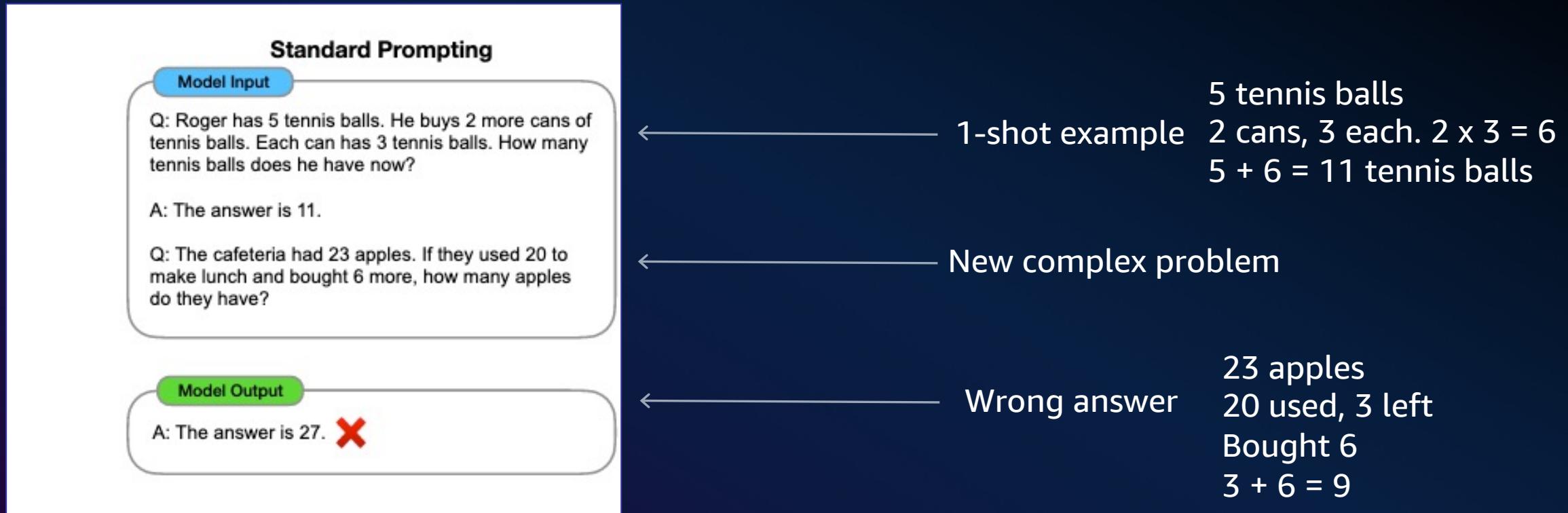
*Enrich context of prompts from internal knowledge repositories*

*Mitigate the effects of hallucination using prompt grounding*

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

# Chain of Thought – CoT

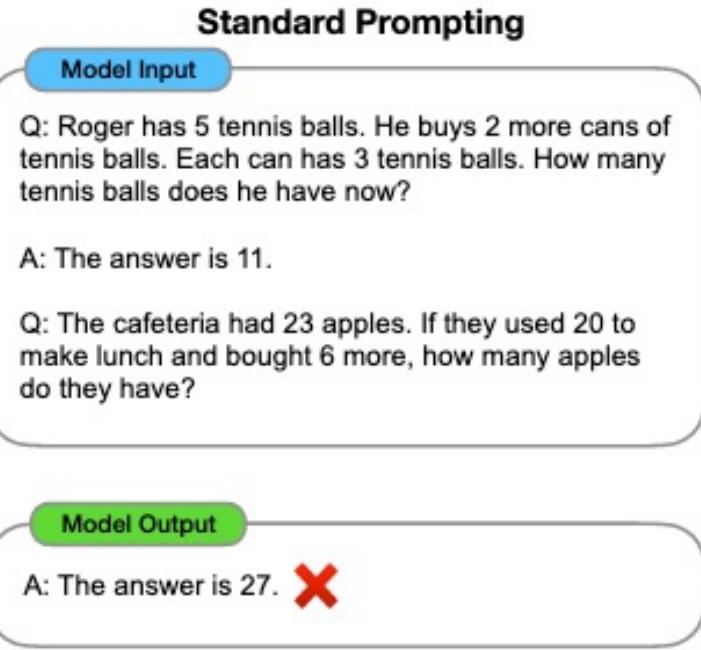
- Explore how series of intermediate steps improve ability of LLMs to perform complex reasoning task



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

# Chain of Thought – CoT (Few-shot-CoT)

(a) Few-shot

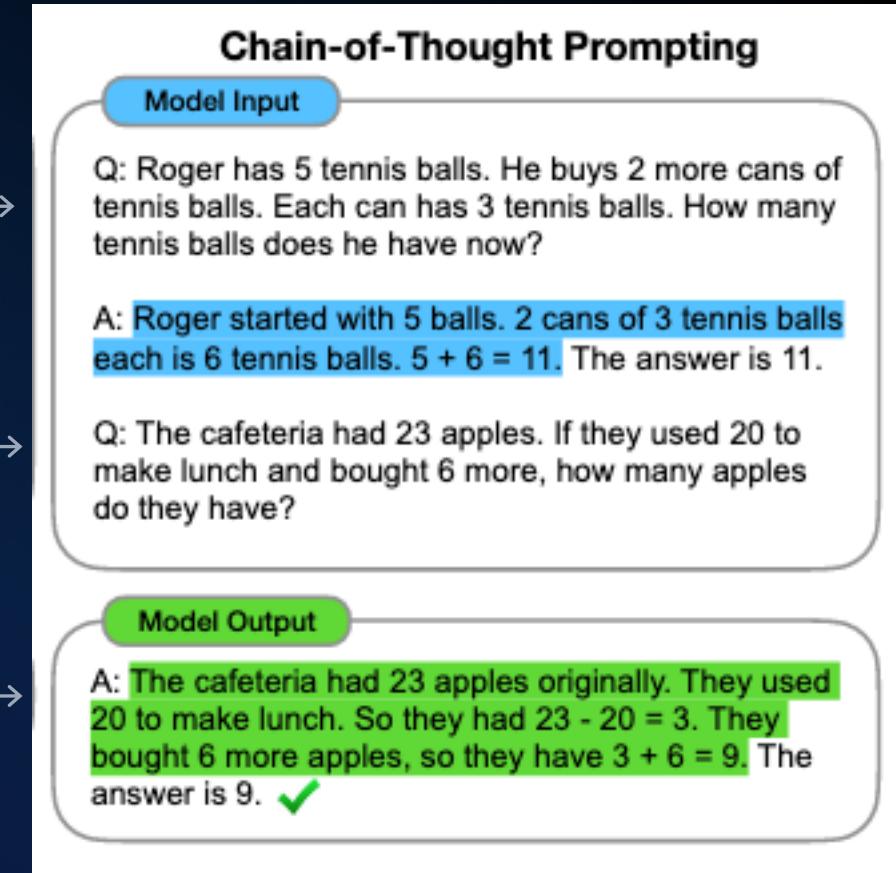


Same 1-shot example

Same new question

Right answer

(b) 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?

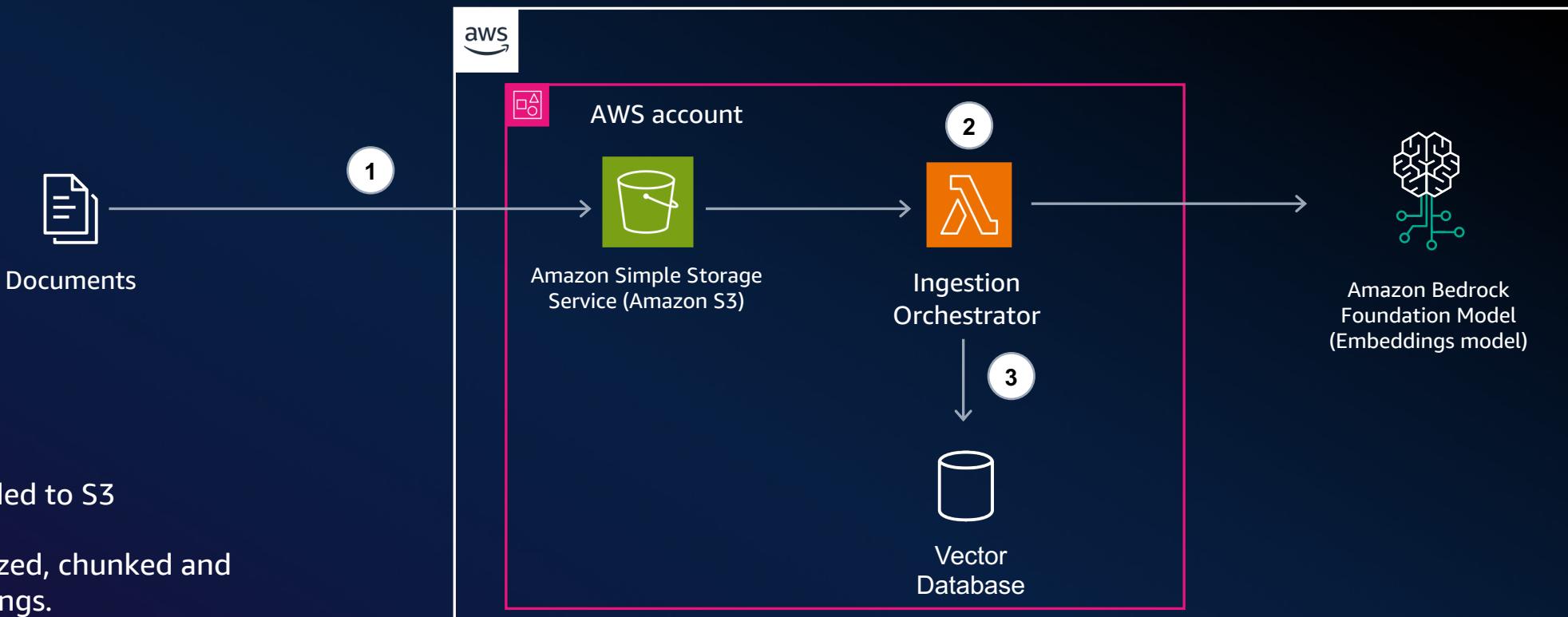
A: **Let's think step by step.**

(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](#)



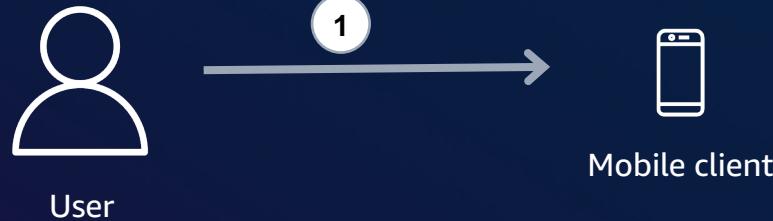
# RAG: System Design – Ingestion of Data for Q+A



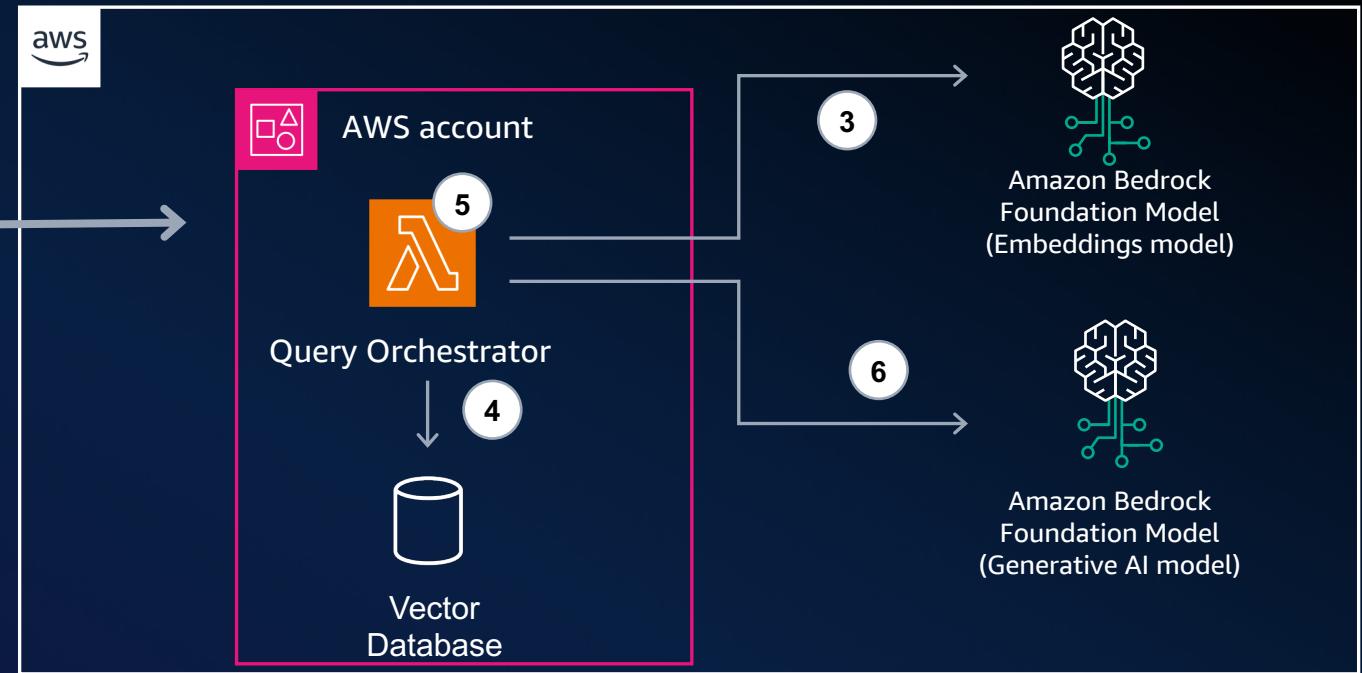
- 1** Documents are uploaded to S3
- 2** Documents are tokenized, chunked and converted to embeddings.
- 3** Embeddings plus metadata stored into a vector database for retrieval process's similarity search.

# RAG: System design – Q+A

What is the return policy?



- 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

Example:

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:”

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>

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>

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

# 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



Claude's response

## Example:

Prompt  
Human: Please write a haiku about {{ANIMAL}}. Use JSON format with the keys as "first\_line", "second\_line", and "third\_line".

Assistant: {

"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*

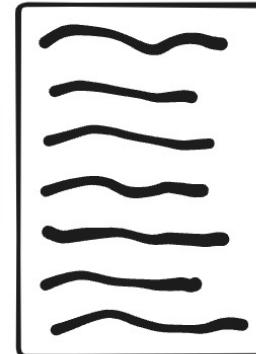
*Search & RAG*

## 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>
```

Function calling prompt



# Advanced prompting techniques

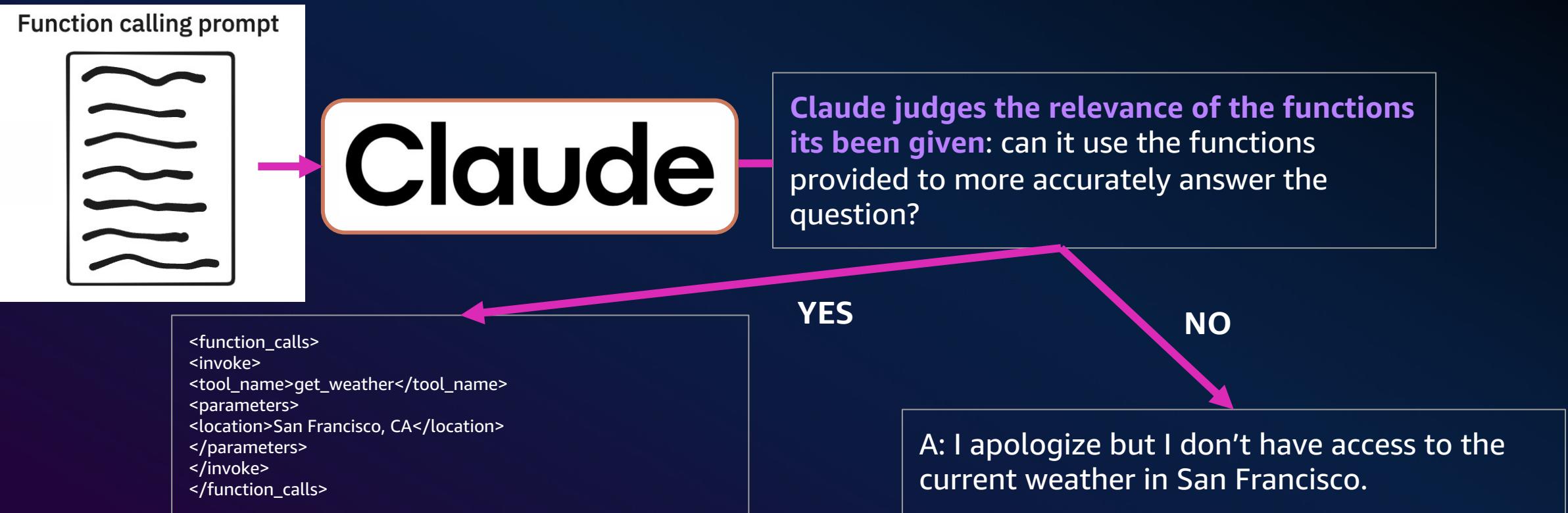
*Chaining prompts*

*100K prompts*

*Agents & function calling / tool use*

*Search & RAG*

## How it works cont'd:



# Advanced prompting techniques

*Chaining prompts*

*100K prompts*

*Agents & function calling / tool use*

*Search & RAG*

## How it works cont'd (if YES):

```
<function_calls> [...] get_weather [...] </function_calls>
```



Claude

```
<function_results>  
[...]  
[68, sunny]  
[...]  
</function_results>>
```



A: The weather right now in San Francisco is sunny with a temperature of 68 degrees

# 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

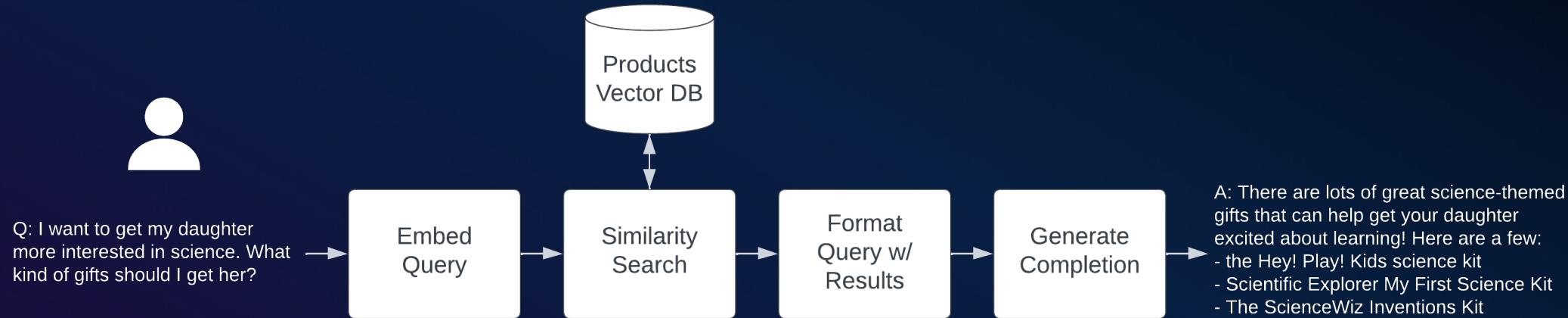
*Chaining prompts*

*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

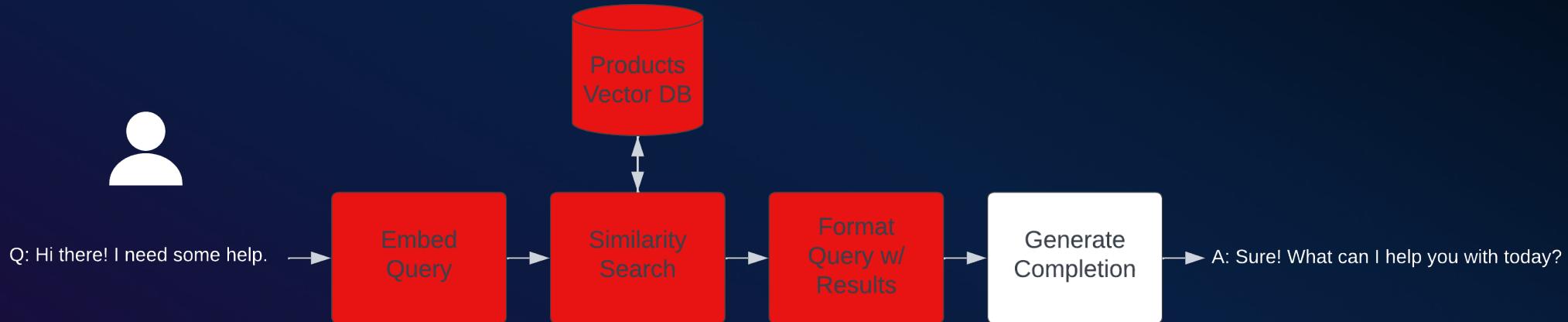
*Chaining prompts*

*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

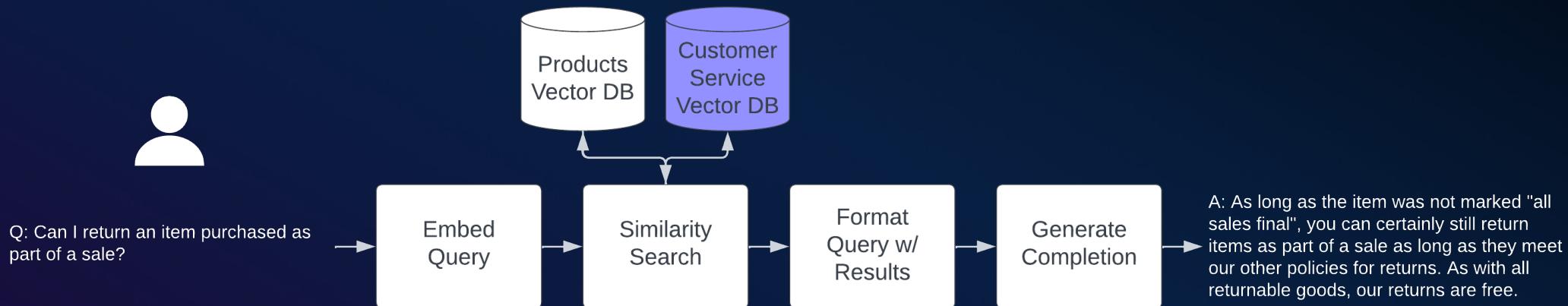
*Chaining prompts*

*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

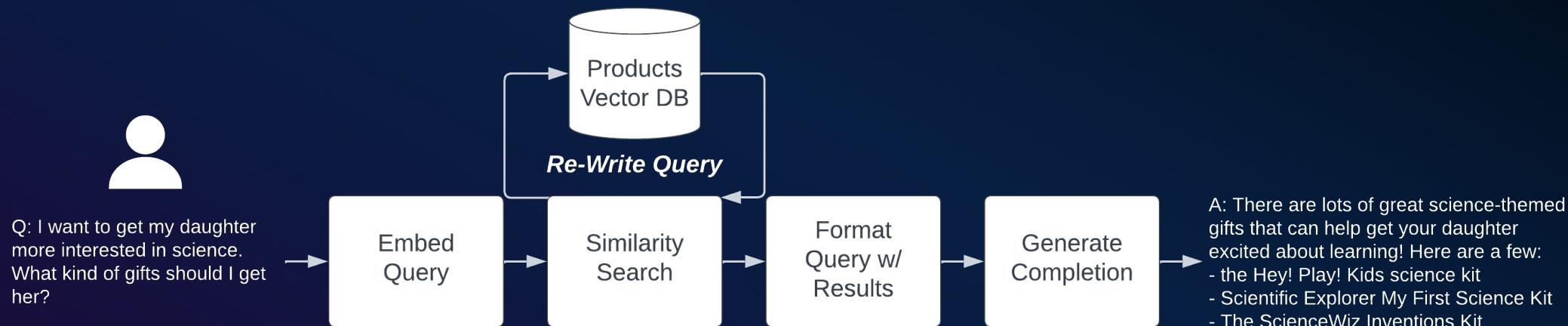
*Chaining prompts*

*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

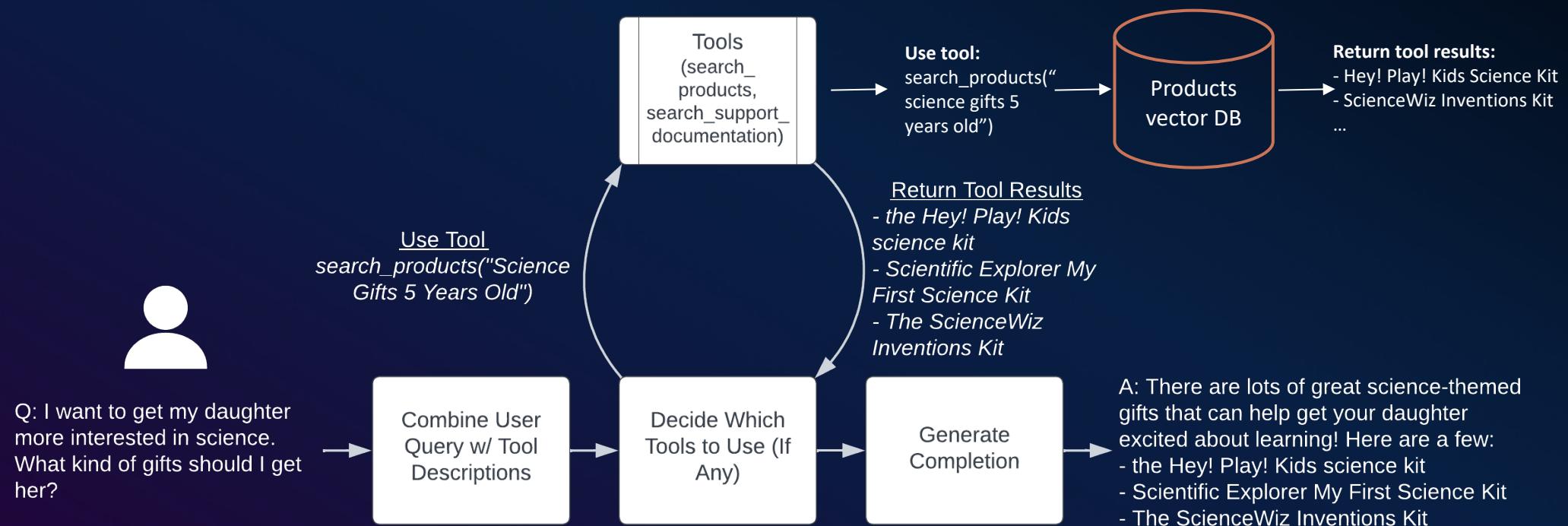
*Chaining prompts*

*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