Tutorial | Build a multimodal knowledge bank for a RAG project

Get started

The Embed documents recipe in Dataiku allows you to augment Large Language Models (LLMs) with specialized internal knowledge from your organization to increase the relevance and accuracy of the model responses.

It can process documents of different formats (`.pdf`, `.docx`, `.pptx`, `.txt` and `.md`) and thus generates a multimodal knowledge bank.

In this tutorial, we'll use the Retrieval Augmented Generation (RAG) approach to enrich an LLM with content from various Dataiku documentation resources to help Dataiku users find relevant answers to their questions.

Objectives

In this tutorial, you will:

- Use the Embed documents recipe to extract the data from selected pages of the Dataiku documentation and vectorize it into a multimodal knowledge bank.
- Create a prompt in the Prompt Studio to evaluate the response from the augmented LLM.

Prerequisites

To use the Embed documents recipe, you'll need:

- Dataiku 13.4 or later.
- An Advanced Analytics Designer or Full Designer user profile.
- A compatible code environment for retrieval augmented models.
- A connection to a supported embedding model for text embedding in the Embed recipe.
- A connection to a supported Generative AI model, which is the model that will be augmented. See LLM connections for details.

Tip

You do not need previous experience with Large Language Models (LLMs), though it would be useful to read the article Concept | Embed recipes and Retrieval Augmented Generation (RAG) before completing this tutorial.

Create the project

Dataiku 13.3+ Dataiku Pre-13.3

1. From the Dataiku Design homepage, click **+ New Project**.
2. Select **Learning projects**.
3. Search for and select **Multimodal Embedding**.
4. Click **Install**.
5. From the project homepage, click **Go to Flow** (or `g` + `f`).

Note

You can also download the starter project from this website and import it as a zip file.

Add the Embed documents recipe

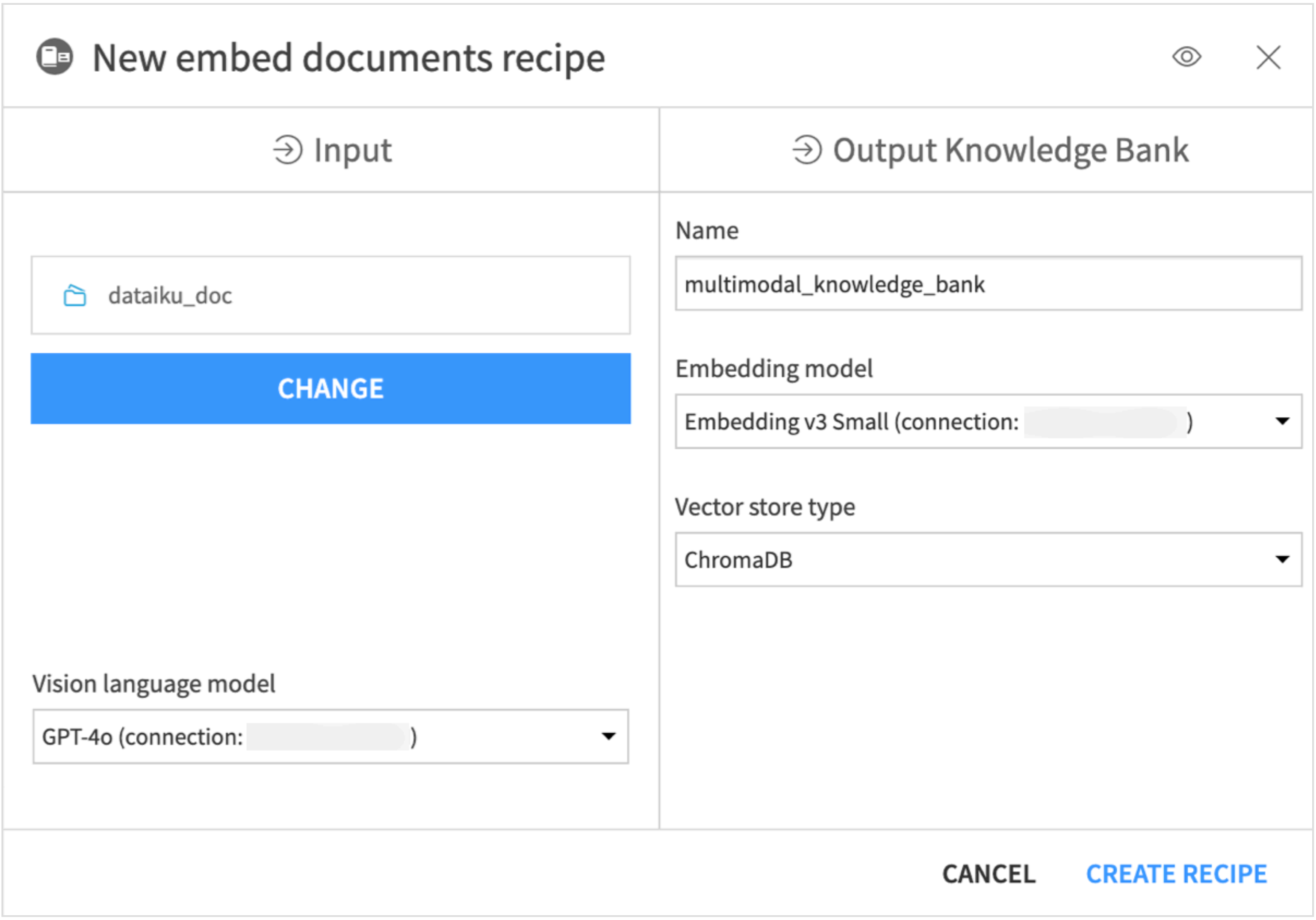
This section will guide you through the creation and configuration of the Embed documents recipe.

Create the Embed documents recipe

Let's first create the Embed document recipe from the data that is stored in the managed folder.

To do so:

1. In the Flow, select the **dataiku_doc** managed folder.
2. Go to the **Actions** tab and click **Embed documents** under the **Visual Recipes** section.
3. In the **New embed documents recipe** window:
   - Keep the selected input.
   - In the **Vision language model** field, select an LLM connection that will be used for the VLM extraction to generate a RAG-friendly summary in order to ensure that the extracted content is concise and more likely to fit within the embedding model's size limits.
   - Name the output knowledge bank `multimodal_knowledge_bank`.
   - In the **Embedding model** field, select a model you can use for text embedding (i.e. text vectorization, which means encoding the semantic information into a numerical representation).
   - Keep the default **ChromaDB** vector store type.
4. Click **Create Recipe**.


New embed documents recipe

Input

dataiku_doc

Vision language model: GPT-4o (connection: [redacted])

Output Knowledge Bank

Name: multimodal_knowledge_bank

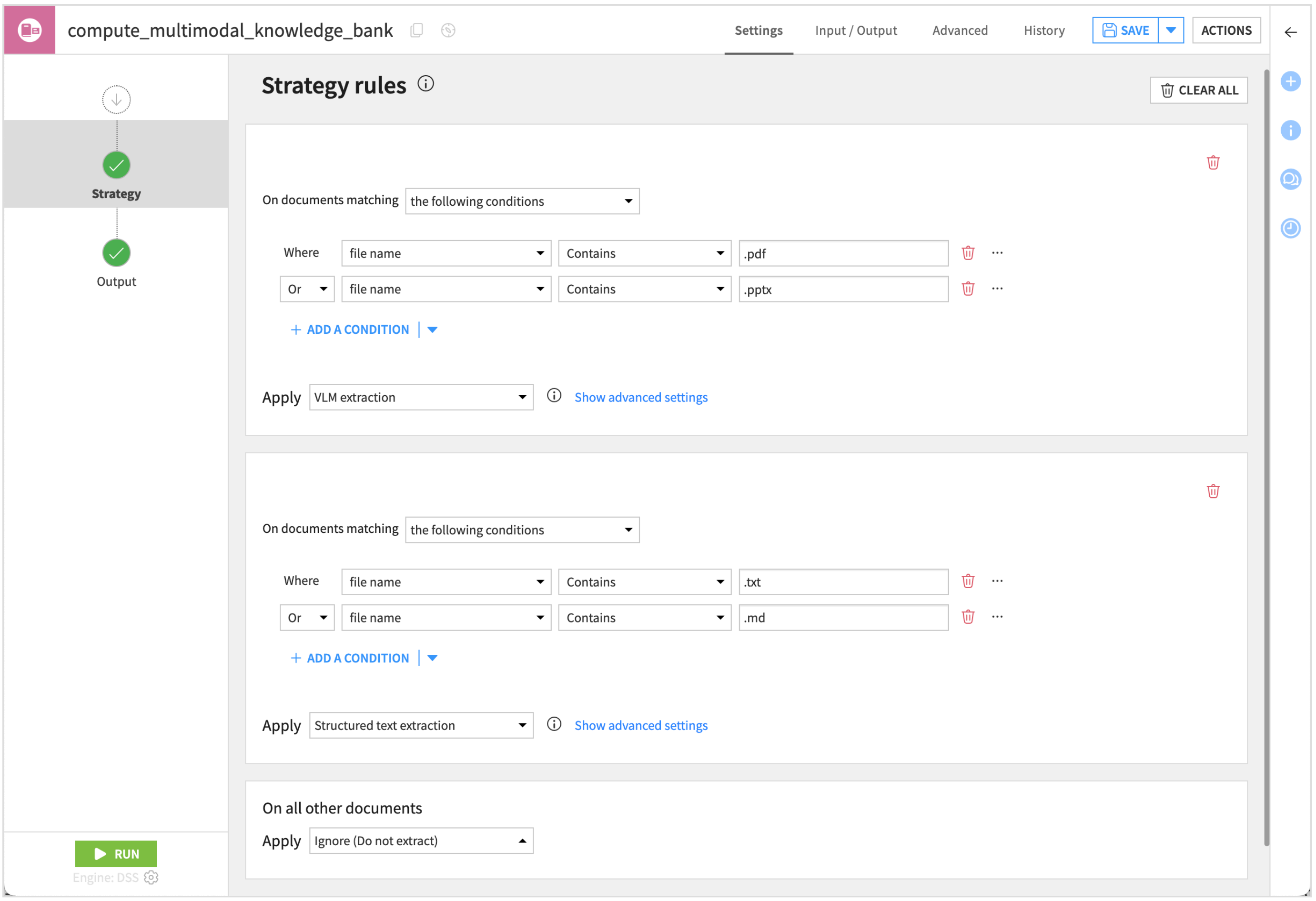
Embedding model: Embedding v3 Small (connection: [redacted])

Vector store type: ChromaDB

Configure the Embed documents recipe

Now, it's time to configure the recipe.

The settings page of the Embed documents recipe allows you to select the extraction strategy to use depending on the file types.



Strategy rules

On documents matching: the following conditions
Where file name Contains .pdf
Or file name Contains .pptx
Apply: VLM extraction

On documents matching: the following conditions
Where file name Contains .txt
Or file name Contains .md
Apply: Structured text extraction

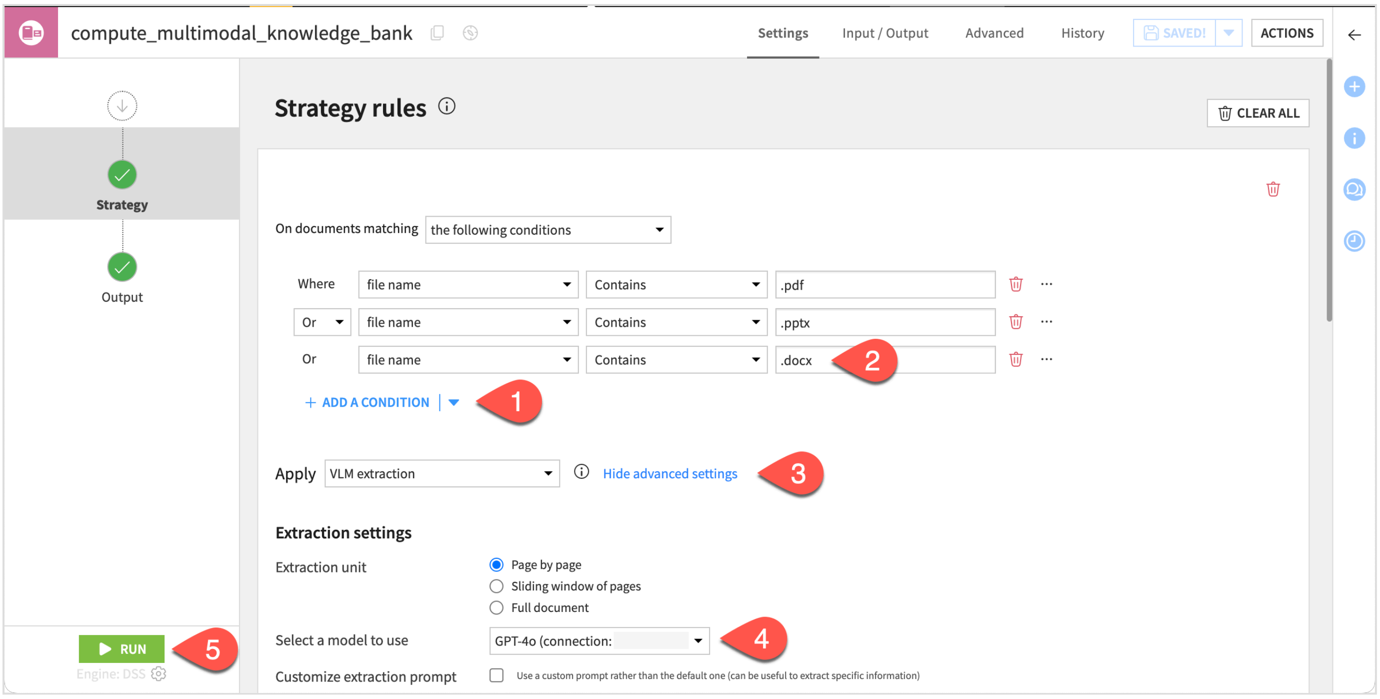
On all other documents
Apply: Ignore (Do not extract)

As shown above, by default, the recipe offers to use the:

- VLM extraction for `.pdf` or `.pptx` files.
- Structured text extraction for `.txt` or `.md` files.

As we also have a Word document (.docx) in our source folder, let's add an extraction rule and manage some advanced settings.

1. In the first block, click **+ Add a Condition**.
2. In the new condition that appears, enter `.docx` to add this extension.
3. Click the **Show advanced settings** to check the extraction rules.
4. In the **Select a model to use** field, ensure that the LLM connection is the one you selected upon creating the recipe, for the summarization of your documents.
5. Leave all other options unchanged and click **Run** to create the knowledge bank, which is the object that stores the output of the text embedding.



Strategy rules

On documents matching: the following conditions

Where	file name	Contains	.pdf
Or	file name	Contains	.pptx
Or	file name	Contains	.docx

+ ADD A CONDITION

Apply: VLM extraction — Hide advanced settings

Extraction settings

Extraction unit: [✓ Page by page] Sliding window of pages / Full document

Select a model to use: GPT-4o (connection:

Customize extraction prompt: ☐ Use a custom prompt rather than the default one (can be useful to extract specific information)

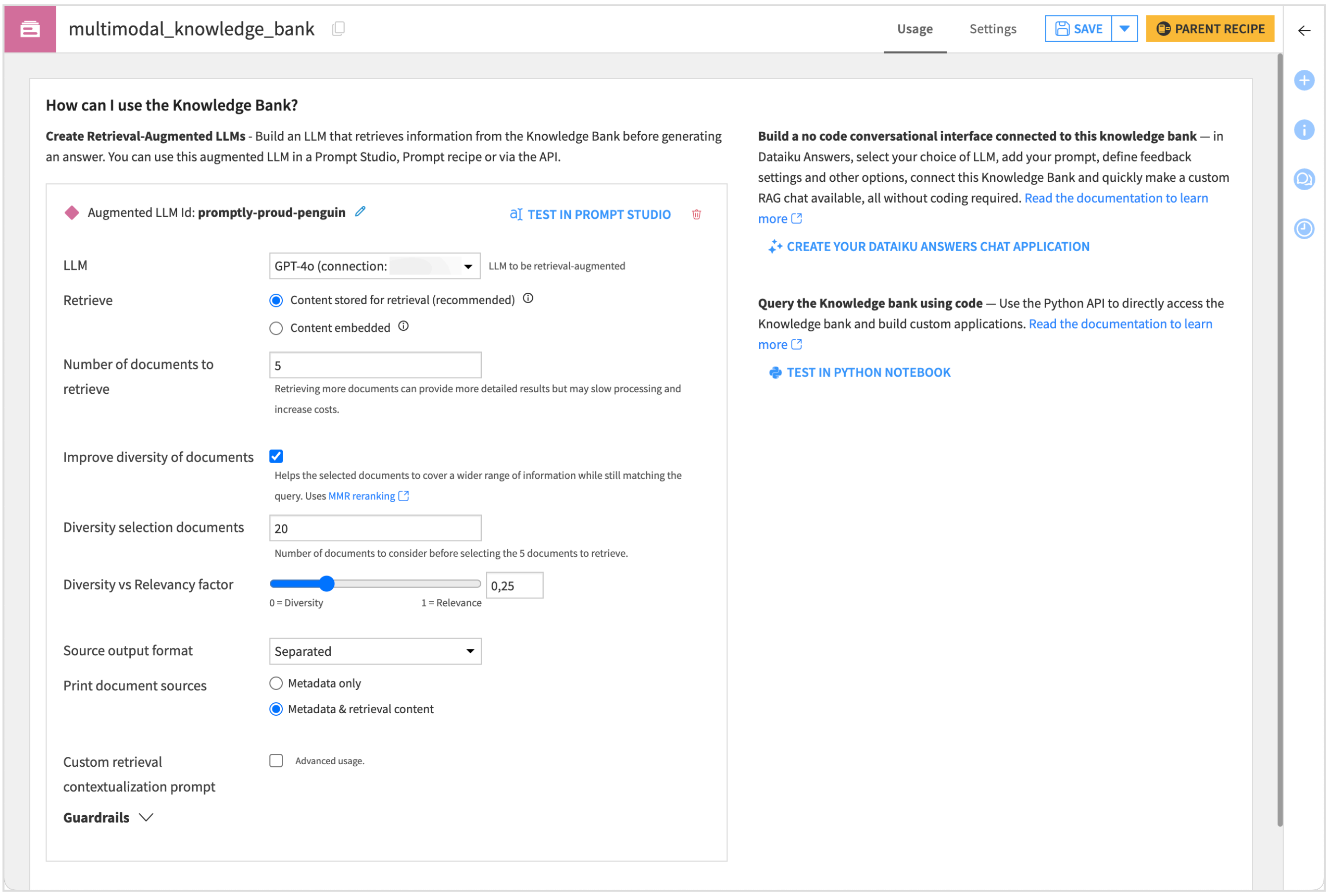
Once the run is over, if you look at the Flow, you'll see two outputs of the Embed document recipe:

Output	Description
dataiku_doc_embedded_images	Includes all images taken from the documents, the content of which has been extracted using the VLM extraction.
multimodal_knowledge_bank	Stores the output of the text embedding.

Configure the knowledge bank

Now that the Embed documents recipe has created the knowledge bank object in the Flow, let's configure it to augment the LLM.

1. From the Flow, double-click **multimodal_knowledge_bank** to open the knowledge bank settings page.
2. In the **Usage** tab, to configure the LLM that we'll augment with the content of the Dataiku documentation, click the **+ Add Augmented LLM** button and fill in the fields as defined below:
   - Leave the generated **Augmented LLM ID** field as is or enter any other ID you wish using the pen icon.
   - In the **LLM** field, select the LLM that you want to augment (here, GPT-4).
   - Set the **Number of documents to retrieve** option to `5`.
   - Enable the **Improve diversity of documents** option and keep the default values for the diversity options.
   - Keep the **Print document sources** option set to **Metadata & retrieval content** to ensure that Dataiku adds to LLM responses details on the sources used to generate each response.
   - Keep the other options unchanged.



How can I use the Knowledge Bank?

Create Retrieval-Augmented LLMs - Build an LLM that retrieves information from the Knowledge Bank before generating an answer. You can use this augmented LLM in a Prompt Studio, Prompt recipe or via the API.

Augmented LLM Id: **promptly-proud-penguin**

LLM: GPT-4o (connection: [redacted] — LLM to be retrieval-augmented

Retrieve: [✓ Content stored for retrieval (recommended)] Content embedded

Number of documents to retrieve: 5
Retrieving more documents can provide more detailed results but may slow processing and increase costs.

Improve diversity of documents: ✓
Helps the selected documents to cover a wider range of information while still matching the query. Uses MMR reranking

Diversity selection documents: 20
Number of documents to consider before selecting the 5 documents to retrieve.

Diversity vs Relevancy factor: 0,25 (0 = Diversity, 1 = Relevance)

Source output format: Separated

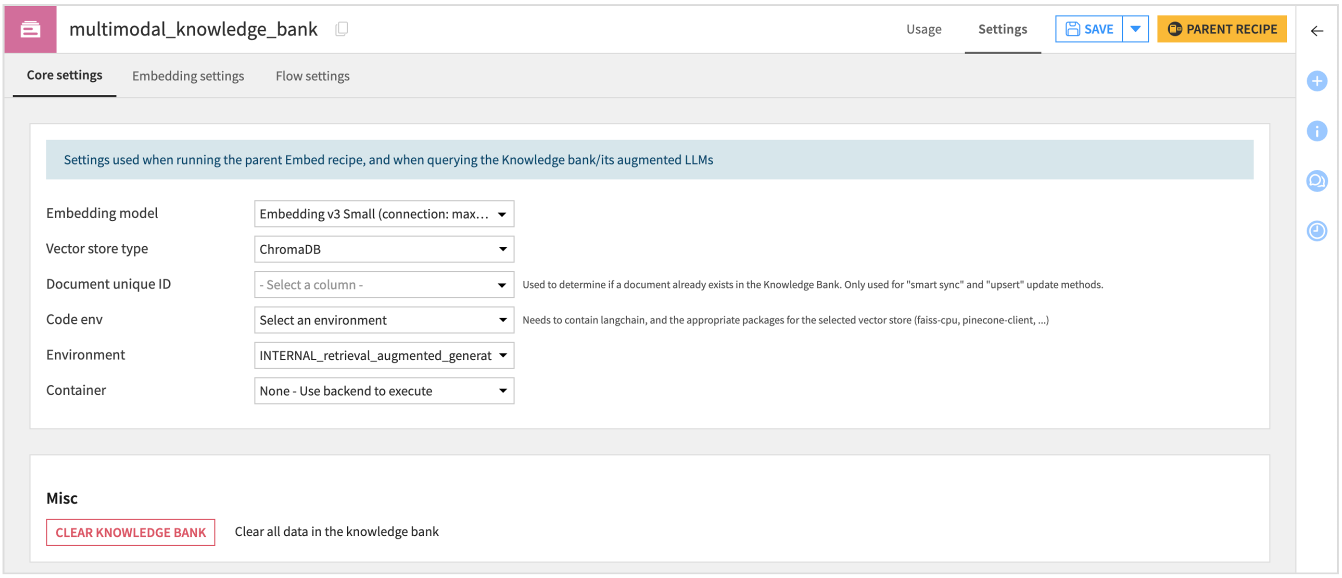
Print document sources: Metadata only [✓ Metadata & retrieval content]

Custom retrieval contextualization prompt: ☐ Advanced usage.

Build a no code conversational interface connected to this knowledge bank — in Dataiku Answers, select your choice of LLM, add your prompt, define feedback settings and other options, connect this Knowledge Bank and quickly make a custom RAG chat available, all without coding required. Read the documentation to learn more

Query the Knowledge bank using code — Use the Python API to directly access the Knowledge bank and build custom applications. Read the documentation to learn more

3. Go to the **Settings** tab and set the **Core settings** subtab as follows:
   - Keep the embedding model and vector store type. These are the ones you selected upon creating the Embed documents recipe.
   - By default, the **Code env** option should be set to **Select an environment** and the **Environment** option to the relevant code environment.
   - Unless you have a specific installation, set the **Container** option to **None - use backend to execute**.



Settings used when running the parent Embed recipe, and when querying the Knowledge bank/its augmented LLMs

Embedding model	Embedding v3 Small (connection: max…	
Vector store type	ChromaDB	
Document unique ID	- Select a column -	Used to determine if a document already exists in the Knowledge Bank. Only used for "smart sync" and "upsert" update methods.
Code env	Select an environment	Needs to contain langchain, and the appropriate packages for the selected vector store (faiss-cpu, pinecone-client, ...)
Environment	INTERNAL_retrieval_augmented_generat	
Container	None - Use backend to execute	

Clear all data in the knowledge bank

4. Click **Save**.
5. Return to the Flow (press `g` + `f`).
6. Double-click the embed recipe and click Run again to take into account the changes in the knowledge bank settings.

> **Note**
>
> In this tutorial, we need to re-run the recipe because we changed the settings in the **Core settings** tab. If you just change the settings of the **Usage** tab, there's no need to run the recipe again. Changes are automatically taken into account.

With this configuration, we augment the LLM with the content of the articles stored initially in the **dataiku_doc** folder and ask that the LLM uses the five top documents among the 20 documents closest to the query to build an answer in plain text.

As we enabled the **Print document sources** option, when testing the augmented LLM in the Prompt Studio (see section below), Dataiku will display the five sources that the model used to generate the answer. Additionally, with the **Include text** option enabled by default, the model will also include the retrieved text from the retrieval column of each source used.

Test the augmented LLM in a Prompt Studio

Now, let's see how the augmented LLM responds to a prompt.

Create the Prompt Studio

The first thing to do is create a Prompt Studio.

1. Go back to the **Usage** tab of the knowledge bank and click **Test in Prompt Studio** next to the augmented LLM.

   > Tip
   >
   > You can also access the Prompt Studios by selecting **Visual ML** () **> Prompt Studios** in the top navigation bar.

2. Give the new studio the name `dataiku_documentation`.
3. Click **Create**. It opens the **Prompt design** page.

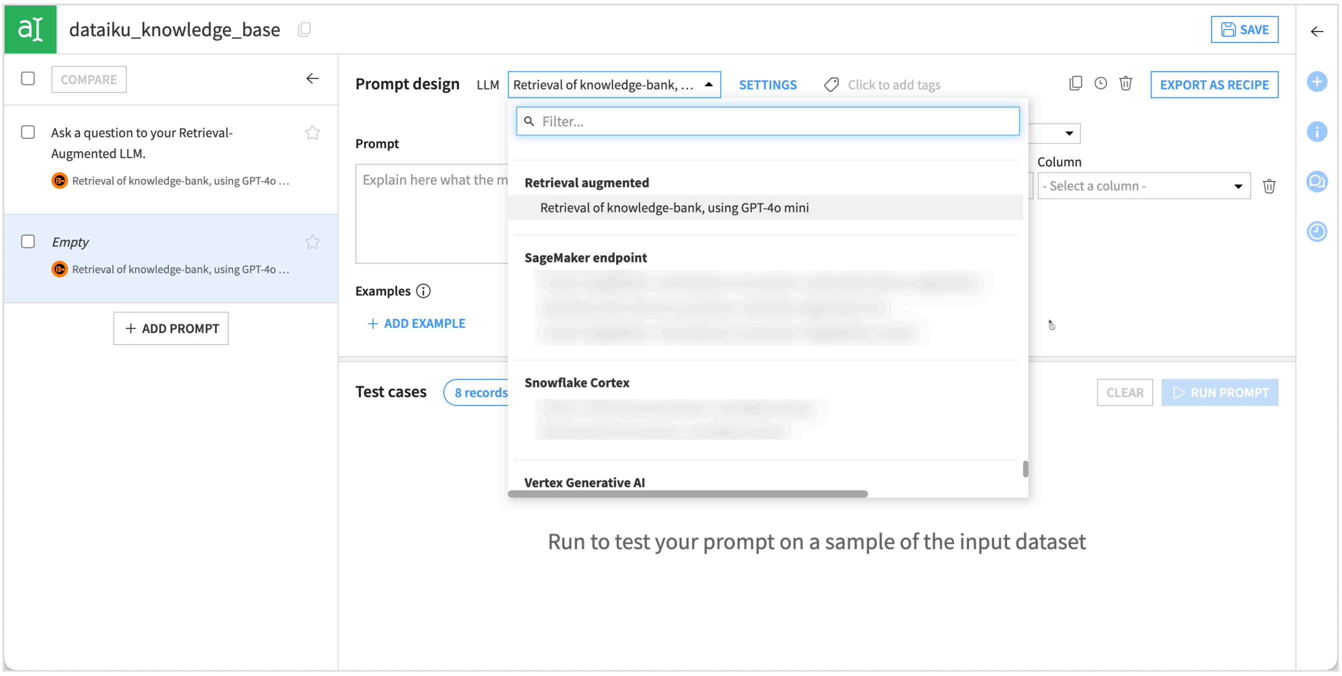
Design a prompt

On the **Prompt design** page, we'll add our prompt text and run a test using the augmented

LLM.

1. In the left panel of the studio, click **+ Add Prompt**.
2. Select **Managed mode > Blank template > Create**.
3. In the **LLM** option, ensure that the augmented LLM in the **Retrieval augmented** section is selected.

> **Note**
>
> The name for an augmented LLM is `Retrieval of <knowledge_bank_id>, using <augmented_model_name>`.
>
> If you augment the same model more than once using the same knowledge bank, the LLM ID you set is added: `Retrieval of <knowledge_bank_id> (id: <llm_id>) using <augmented_model_name>`.



4. In the **Prompt** field, copy and paste the following prompt. Use the **Copy** button at the right of the block for easier copying.

```
You're an expert in Dataiku and rely on the knowledge from the Dataiku documentation.
```

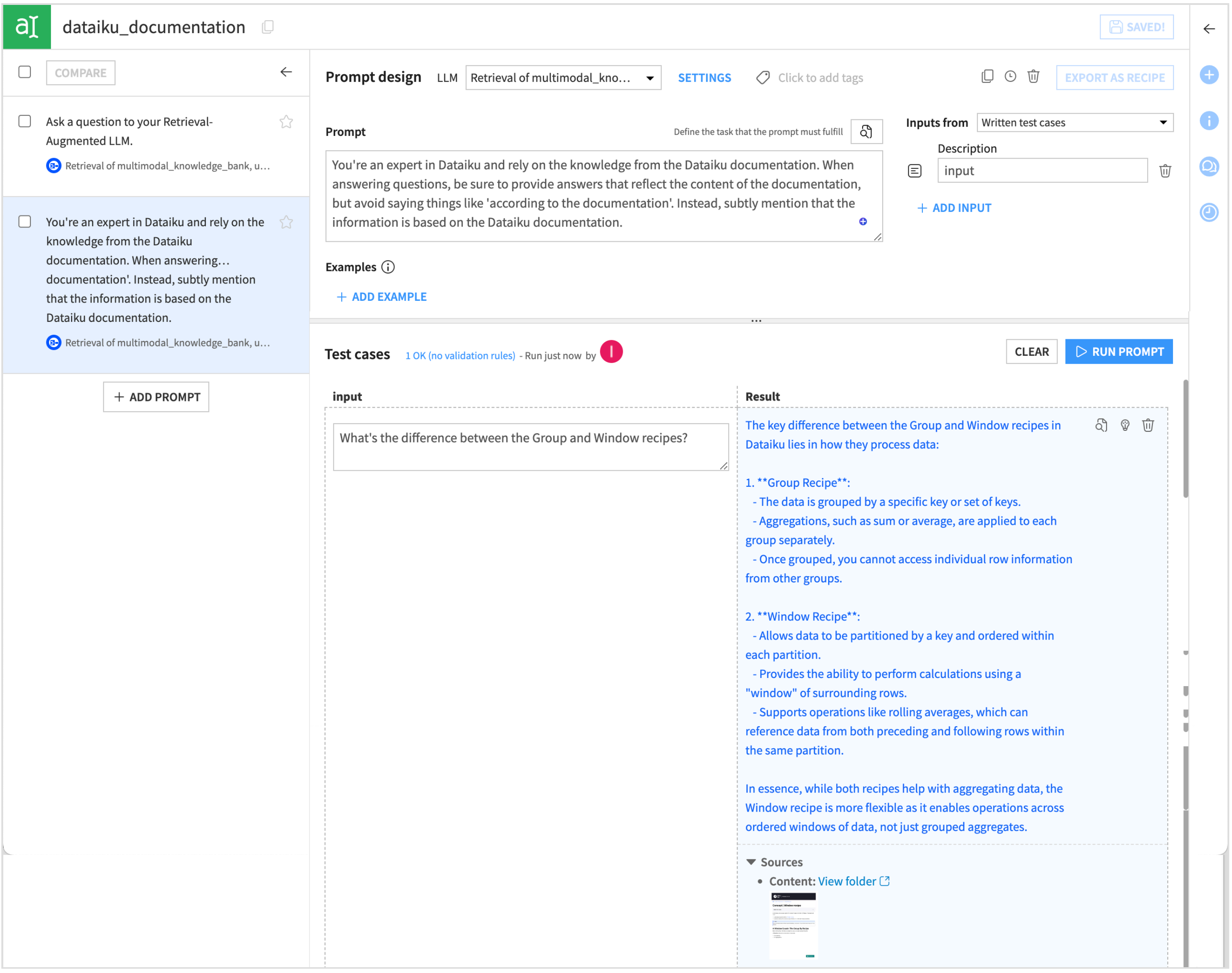
3. On the right, in the **Inputs from** dropdown menu, select **Written test cases**.

> **Note**
>
> Feel free to rename the Description field. Instead of input, you can enter `Question` for instance. This changes the column header under **Test cases**.

4. Under **Test cases**, click **+ Add Test Case** to add a test to gauge how our model runs the prompt as it is.
5. Copy and paste the following text into the **input** box:

```
What's the difference between the Group and Window recipes?
```

6. Click **Run Prompt** to pass the prompt and test case to your selected model.



dataiku_documentation
SAVED!
COMPARE
Ask a question to your Retrieval-Augmented LLM.
Retrieval of multimodal_knowledge_bank, u…
You're an expert in Dataiku and rely on the knowledge from the Dataiku documentation. When answering… documentation'. Instead, subtly mention that the information is based on the Dataiku documentation.
Retrieval of multimodal_knowledge_bank, u…
ADD PROMPT
Prompt design
LLM
Retrieval of multimodal_kno…
SETTINGS
Click to add tags
EXPORT AS RECIPE
Prompt
Define the task that the prompt must fulfill
Inputs from
Written test cases
Description
input
ADD INPUT
Examples
ADD EXAMPLE
Test cases
1 OK (no validation rules) - Run just now by
CLEAR
RUN PROMPT
input
Result

Prompt: You're an expert in Dataiku and rely on the knowledge from the Dataiku documentation. When answering questions, be sure to provide answers that reflect the content of the documentation, but avoid saying things like 'according to the documentation'. Instead, subtly mention that the information is based on the Dataiku documentation.

input: What's the difference between the Group and Window recipes?

Result:

The key difference between the Group and Window recipes in Dataiku lies in how they process data:

1. **Group Recipe**:
 - The data is grouped by a specific key or set of keys.
 - Aggregations, such as sum or average, are applied to each group separately.
 - Once grouped, you cannot access individual row information from other groups.

2. **Window Recipe**:
 - Allows data to be partitioned by a key and ordered within each partition.
 - Provides the ability to perform calculations using a "window" of surrounding rows.
 - Supports operations like rolling averages, which can reference data from both preceding and following rows within the same partition.

In essence, while both recipes help with aggregating data, the Window recipe is more flexible as it enables operations across ordered windows of data, not just grouped aggregates.

Sources
- **Content:** View folder

The results may differ from those above, as LLMs do not generate the same response every time.

Concretely, here's what happened upon running the test:

1. Based on the initial prompt you enter, the knowledge bank identifies five chunks of text that are similar to the prompt. You see them in the **Sources** section at the bottom of the response.

   Note

   Why five? This is because we asked to retrieve only five documents in the **Documents to retrieve** option of the **Usage** tab of the knowledge bank.

2. These five text chunks are fetched from the knowledge bank and the text from their retrieval column is added to the prompt.
3. The LLM generates a response based on this augmented prompt.
4. Dataiku adds the metadata (here, the original article URLs and raw content) in the **Sources** section at the bottom of the response.

What's next?

Now that you know how to augment an LLM with your specific knowledge, you could:

- Create a chatbot using Dataiku Answers.
- Create a dataset with some questions to use it for test cases in a Prompt Studio, then create a Prompt recipe from it.

See also

For more information:

- On the Embed documents recipe and the RAG approach, see the Concept | Embed recipes and Retrieval Augmented Generation (RAG) article. (WIP)
- On LLM evaluation, see the Tutorial | LLM evaluation article.
- On guardrails, see the content moderation and sensitive data management.