Reasoning: Load the json file into a pandas DataFrame.

```
import pandas as pd
try:
    df_hotels = pd.read_json('filtered_hotels.json').head(2)
    display(df_hotels.head())
    display(df_hotels.info())
except FileNotFoundError:
    print("Error: 'filtered_hotels.json' not found.")
    df_hotels = None
except ValueError as e:
    print(f"Error: Could not parse JSON file: {e}")
    df_hotels = None
except Exception as e:
    print(f"An unexpected error occurred: {e}")
    df_hotels = None
```

	hotel_id	name	location	price	rating	features	nearby_beaches	comments
0	1	sousse pearl marriott resort spa	Sousse	152	Avec une note de 8,2	['4 piscines', 'Navette aéroport', 'Chambres n...	['Plage de Boujaafar\n500 m', 'Plage de Bhar E...	[{'titre': 'Exceptionnel', 'commentaire': 'J'a...
1	2	hotel medina	Sousse	75	Avec une note de 7,7	['Chambres non-fumeurs', 'Parking', 'Connexion...	['Plage de Bhar Ezzebla\n750 m', 'Plage de Bou...	[{'titre': 'Exceptionnel', 'commentaire': 'Bel...

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2 entries, 0 to 1
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   hotel_id        2 non-null      int64
 1   name            2 non-null      object
 2   location        2 non-null      object
 3   price           2 non-null      int64
 4   rating          2 non-null      object
 5   features        2 non-null      object
 6   nearby_beaches  2 non-null      object
 7   comments        2 non-null      object
dtypes: int64(2), object(6)
memory usage: 260.0+ bytes
```

Data preparation

Subtask:

Prepare the data for embedding generation.

Note wl Prix yekhdmo

```
if df_hotels is not None and 'nearby_beaches' in df_hotels.columns:
  for index, row in df_hotels.iterrows():
    if isinstance(row['nearby_beaches'], str):
      df_hotels.at[index, 'nearby_beaches'] = row['nearby_beaches'].replace('\\n', ' ')
  print(df_hotels["nearby_beaches"])
else:
  print("DataFrame or 'nearby_beaches' column not found.")
df_hotels["nearby_beaches"]
```

```
0    ['Plage de Boujaafar 500 m', 'Plage de Bhar Ez...
1    ['Plage de Bhar Ezzebla 750 m', 'Plage de Bouj...
Name: nearby_beaches, dtype: object
```

	nearby_beaches
0	['Plage de Boujaafar 500 m', 'Plage de Bhar Ez...
1	['Plage de Bhar Ezzebla 750 m', 'Plage de Bouj...

dtype: object

```
# df_hotels['text'] = (
#     df_hotels['name'].fillna('Nom inconnu') + ', ' +
#     df_hotels['location'].fillna('Localisation inconnue') + ', ' +
#     df_hotels['price'].fillna('Prix non renseigné').astype(str) + ' DT, ' +
#     df_hotels['rating'].fillna('Note non disponible').astype(str)
# )


# # Inspect the 'text' column
# display(df_hotels.head())
# display(df_hotels['text'].unique())


import ast
# clean the features and beaches
def clean_list_field(field_value):
    try:
        items = ast.literal_eval(field_value)
        if isinstance(items, list):
            cleaned = list(dict.fromkeys([item.replace('\\n', ' ').strip().lower() for item in items if isinstance(item, s
            return ', '.join(cleaned)
    except Exception:
        pass
    return 'non renseigné'
# hezz ken l comments
def extract_individual_comments(row):
    """
    Extrait et nettoie les commentaires dans une liste séparée,
    supprimant les '\n' et ne gardant que le contenu du commentaire.
    """
    comments = row['comments']
    if isinstance(comments, list):
        # Nettoyer chaque commentaire et enlever les '\n' (saute de ligne)
        cleaned_comments = [c.get('commentaire', '').replace("\n", " ") for c in comments]
        return cleaned_comments
    else:
        return ['Aucun commentaire disponible']

# Appliquer cette fonction à chaque ligne du DataFrame pour extraire les commentaires
df_hotels['individual_comments'] = df_hotels.apply(extract_individual_comments, axis=1)

def generate_text(row):
    name = row.get('name', 'nom inconnu')
    location = row.get('location', 'localisation inconnue')
    price = row.get('price', 'prix non renseigné')
    rating = row.get('rating', 'note non disponible')

    features = clean_list_field(row.get('features', ''))
    beaches = clean_list_field(row.get('nearby_beaches', ''))

      # les commentaires f liste whdhom donc bch naamloulhom jointure
    # Traitement des commentaires
    comments = row['individual_comments']
    if comments:
    # Afficher tous les commentaires individuellement
      cleaned_comments = ' | '.join([f"{c}" for c in comments])
    else:
      cleaned_comments = 'Aucun commentaire disponible'

    return (
        f"Nom de l'hôtel : {name}.\n"
        f"Localisation : {location}.\n"
        f"Prix : {price} TND par nuit.\n"
        f"Note : {rating}.\n"
        f"Caractéristiques : {features}.\n"
        f"Plages à proximité : {beaches}.\n"
        f"Avis clients : {cleaned_comments}."
    )

df_hotels['text'] = df_hotels.apply(generate_text, axis=1)
```

Feature engineering

Subtask:

Generate embeddings for the 'text' column in the `df_hotels` DataFrame using Hugging Face embeddings.

```
!pip install langchain langchain-community pypdf chromadb -q
!pip install langchain_groq -q
!pip install -U langchain-huggingface -q
!pip install -U langchain-chroma -q
!pip install gradio -q
```

```
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 67.3/67.3 kB 4.7 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.5/2.5 MB 52.2 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 303.4/303.4 kB 24.3 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 18.9/18.9 MB 38.4 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 94.9/94.9 kB 9.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 284.2/284.2 kB 26.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.0/2.0 MB 88.8 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 101.6/101.6 kB 10.5 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 16.4/16.4 MB 114.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 55.9/55.9 kB 5.5 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 194.9/194.9 kB 20.1 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 65.8/65.8 kB 6.3 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 118.9/118.9 kB 12.3 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 92.0/92.0 kB 10.1 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 44.4/44.4 kB 4.3 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.5/62.5 kB 6.4 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 459.8/459.8 kB 41.3 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 50.9/50.9 kB 5.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 71.5/71.5 kB 7.7 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.0/4.0 MB 102.3 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 454.8/454.8 kB 39.7 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 46.0/46.0 kB 606.8 kB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 86.8/86.8 kB 8.9 MB/s eta 0:00:00
  Building wheel for pypika (pyproject.toml) ... done
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.5/127.5 kB 5.8 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 437.7/437.7 kB 15.4 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 363.4/363.4 MB 3.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.8/13.8 MB 102.4 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 24.6/24.6 MB 89.1 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 883.7/883.7 kB 57.1 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 1.3 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 5.7 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 12.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 7.5 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 5.5 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 80.0 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 611.1/611.1 kB 14.4 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.4/2.4 MB 82.2 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 54.1/54.1 MB 16.3 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 322.9/322.9 kB 28.9 MB/s eta 0:00:00
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.5/11.5 MB 133.7 MB/s eta 0:00:00
```

```
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain_huggingface import HuggingFaceEmbeddings
from langchain_community.vectorstores import Chroma
from langchain.chains import RetrievalQA
from langchain_groq import ChatGroq
from langchain.prompts import PromptTemplate
from langchain.schema import Document

import os
import gradio as gr
import json
```

```
!pip install language_tool_python
```

```
Collecting language_tool_python
  Downloading language_tool_python-2.9.3-py3-none-any.whl.metadata (54 kB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 54.7/54.7 kB 3.6 MB/s eta 0:00:00
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from language_tool_python) (2.32.3
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from language_tool_python) (4.67.1)
Requirement already satisfied: psutil in /usr/local/lib/python3.11/dist-packages (from language_tool_python) (5.9.5)
Requirement already satisfied: toml in /usr/local/lib/python3.11/dist-packages (from language_tool_python) (0.10.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->lar
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->language_tool_p
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->language_
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->language_
Downloading language_tool_python-2.9.3-py3-none-any.whl (55 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 55.1/55.1 kB 4.3 MB/s eta 0:00:00
Installing collected packages: language_tool_python
Successfully installed language_tool_python-2.9.3
```

pour corriger le probleme des reponses j'ai changer le model name vers e5-large khtr yhezz akther tokens hata l 32k ama yab9a 9rabet 45 min bch ylanci

```
# Initialize the Hugging Face embeddings model
#embedding_function = HuggingFaceEmbeddings(model_name="sentence-transformers/all-MiniLM-L6-v2")

# switching to another model with more tokens
embedding_function = HuggingFaceEmbeddings(model_name="intfloat/e5-large-v2")

# Generate embeddings for the 'text' column
hotel_embeddings = embedding_function.embed_documents(df_hotels['text'].tolist())

# Print the shape of the embeddings
print(f"Shape of the embeddings: {len(hotel_embeddings)}, {len(hotel_embeddings[0])}")
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
modules.json: 100%                    387/387 [00:00<00:00, 37.6kB/s]
README.md: 100%                       67.8k/67.8k [00:00<00:00, 5.25MB/s]
sentence_bert_config.json: 100%       57.0/57.0 [00:00<00:00, 5.04kB/s]
config.json: 100%                     616/616 [00:00<00:00, 51.7kB/s]
model.safetensors: 100%               1.34G/1.34G [00:08<00:00, 188MB/s]
tokenizer_config.json: 100%           314/314 [00:00<00:00, 33.3kB/s]
vocab.txt: 100%                       232k/232k [00:00<00:00, 3.34MB/s]
tokenizer.json: 100%                  711k/711k [00:00<00:00, 4.97MB/s]
special_tokens_map.json: 100%         125/125 [00:00<00:00, 13.4kB/s]
config.json: 100%                     201/201 [00:00<00:00, 21.5kB/s]
Shape of the embeddings: 2, 1024
```

9a3ed nfasakh fl chroma db f kol mara bch yaawd ysavi les index jdod , solution lel caractéristique heya eni zedet fl chunk size wl overlap

kif zedet chunk brcha wala ykhalwedh w ya9rach f kol chy

```
import shutil
shutil.rmtree("chroma_db", ignore_errors=True)


from langchain_community.vectorstores import Chroma
from langchain.schema import Document
from langchain.text_splitter import RecursiveCharacterTextSplitter

# Split the text into chunks
splitter = RecursiveCharacterTextSplitter(chunk_size=1000, chunk_overlap=300)
documents = [Document(page_content=t) for t in df_hotels['text'].tolist()]
```

```
split_docs = splitter.split_documents(documents)

# Persisted vectorstore
persist_directory = "chroma_db"


if os.path.exists(persist_directory):
    print("Chargement de la base Chroma existante...")
    vectorstore = Chroma(persist_directory=persist_directory, embedding_function=embedding_function)
else:
    print("Création d'une nouvelle base Chroma...")
    vectorstore = Chroma.from_documents(
    documents=split_docs,
    embedding=embedding_function,
    persist_directory=persist_directory,
    collection_name="hotels",
    )
vectorstore.persist()

# Now reload the persisted vectorstore
vectorstore = Chroma(
    embedding_function=embedding_function,
    persist_directory=persist_directory,
    collection_name="hotels"
)
```

```
Création d'une nouvelle base Chroma...
<ipython-input-14-23a6dcc2309f>:25: LangChainDeprecationWarning: Since Chroma 0.4.x the manual persistence method is r
  vectorstore.persist()
<ipython-input-14-23a6dcc2309f>:28: LangChainDeprecationWarning: The class `Chroma` was deprecated in LangChain 0.2.9
  vectorstore = Chroma(
```

test fi chroma DB

```
print(f"Nombre de documents indexés : {vectorstore._collection.count()}")
for doc in split_docs:
    if "medina" in doc.page_content.lower():
        print(doc.page_content)
```

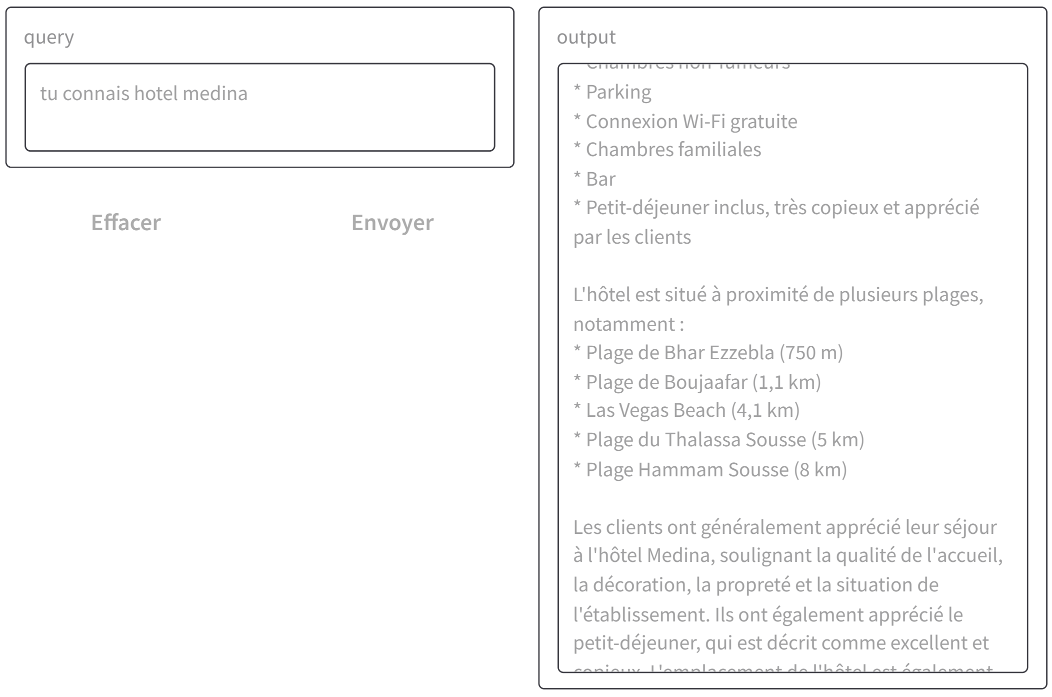
```
Nombre de documents indexés : 7
Nom de l'hôtel : hotel medina.
Localisation : Sousse.
Prix : 75 TND par nuit.
Note : Avec une note de 7,7.
Caractéristiques : chambres non-fumeurs, parking, connexion wi-fi gratuite, chambres familiales, bar, très bon petit-c
Plages à proximité : plage de bhar ezzebla 750 m, plage de boujaafar 1,1 km, las vegas beach 4,1 km, plage du thalassa
Avis clients : Bel hôtel proche plage et medina | L'emplacement est impeccable. La chambre était propore. Le petit déj
L' accueil est familial et chaleureux. Le petit déjeuner est excellent. A peine sorti, on est à la Grande Mosquée et a
client et très prévenante. Nous avons aimé notre séjour dans cet hôtel. | Hôtel à l'entrée de la médina. Chambre famil
```

```
# Initialize the ChatGroq LLM
#llm = ChatGroq(model="llama-3.3-70b-versatile", api_key="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")

# Create a RetrievalQA chain#

#qa_chain = RetrievalQA.from_chain_type(
#    llm=llm, retriever=vectorstore.as_retriever()
#)
```

j'ai testé le mixtral-8x7b mais il donnne un errur

```
llm = ChatGroq(model="llama-3.3-70b-versatile", api_key="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")

prompt_template = PromptTemplate(
   template=(
        "Tu es un assistant touristique tunisien spécialisé dans les recommandations d'hôtels.\n"
        "Utilise uniquement les informations fournies dans le contexte ci-dessous.\n"
        "**N'invente jamais.** Si une information est absente, indique : 'Non renseigné'.\n\n"
        "Contexte : {context}\n"
        "Question du client : {question}\n\n"
        "Réponse détaillée:"
   ),
   input_variables=["context", "question"]
)

qa_chain = RetrievalQA.from_chain_type(
   llm=llm,
   chain_type="stuff",
   retriever=vectorstore.as_retriever(),
   return_source_documents=True,
   chain_type_kwargs={"prompt": prompt_template}
)


def chatbot(query):
    result = qa_chain({"query": query})
    return result['result']

iface = gr.Interface(
    fn=chatbot,
    inputs=gr.Textbox(lines=2, placeholder="Enter your question here..."),
    outputs="text",
    title="Hotel RAG Chatbot",
)

iface.launch()
```

```
It looks like you are running Gradio on a hosted a Jupyter notebook. For the Gradio app to work, sharing must be enabl

Colab notebook detected. To show errors in colab notebook, set debug=True in launch()
* Running on public URL: https://2e35029f4f2aa1e23a.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal
```



tu connais hotel medina

* Parking
* Connexion Wi-Fi gratuite
* Chambres familiales
* Bar
* Petit-déjeuner inclus, très copieux et apprécié par les clients

L'hôtel est situé à proximité de plusieurs plages, notamment :
* Plage de Bhar Ezzebla (750 m)
* Plage de Boujaafar (1,1 km)
* Las Vegas Beach (4,1 km)
* Plage du Thalassa Sousse (5 km)
* Plage Hammam Sousse (8 km)

Les clients ont généralement apprécié leur séjour à l'hôtel Medina, soulignant la qualité de l'accueil, la décoration, la propreté et la situation de l'établissement. Ils ont également apprécié le petit-déjeuner, qui est décrit comme excellent et