Package Install

[ ] ↳ *숨겨진 셀 6개*

실행 데모

```
from langchain_groq import ChatGroq
from langchain_core.prompts import ChatPromptTemplate

# ChatGroq 모델 초기화
llm = ChatGroq(
    model="gemma2-9b-it",
    temperature=0.7,
    max_tokens=300,
    api_key="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"
)
```

```
prompt = ChatPromptTemplate.from_messages([
    ("system", "당신은 친절하고 유익한 AI 조수입니다. 한국의 역사와 문화에 대해 잘 알고 있습니다."
    ("human", "{question}")
])

chain = prompt | llm

# 질문 리스트
questions = [
    "한글의 창제 원리는 무엇인가요?",
    "김치의 역사와 문화적 중요성에 대해 설명해주세요.",
    "조선시대의 과거 제도에 대해 간단히 설명해주세요."
]

# 각 질문에 대한 답변 생성
for question in questions:
    response = chain.invoke({"question": question})
    print(f"질문: {question}")
    print(f"답변: {response.content}₩n")
```

```
질문: 한글의 창제 원리는 무엇인가요?
답변: 네, 한국의 역사와 문화에 대해 돕고 싶습니다!

한글의 창제 원리는 다음과 같이 요약될 수 있습니다.

* **자연의 소리에 기반:** 세종대왕은 자연의 소리, 특히 인간의 발음을 섬세하게 관찰하여 소리를

* **음성 기호와 자음, 모음 활용:**

    * 자음은 **음성의 발음 방식**을 나타내며, 입 모양, 호흡, 혀의 위치 등을 기반으로 만들었습
    * 모음은 **입 모양**을 나타내며,
      * 열린 구강, 닫힌 구강, 혀의 위치 등을 기반으로 만들었습니다.
    * 자음과 모음을 조합하여 글자를 만들었습니다.
```

* **다양한 발음 표현:** 한글은 자음과 모음을 다양하게 조합하여 소리의 변화를 표현할 수 있도록

* **학습 및 사용의 용이성:** 한글 자체의 형태가 매우 명확하고 논리적이기 때문에 학습과 사용이

세종대왕은 사람들이 쉽게 글을 읽

질문: 김치의 역사와 문화적 중요성에 대해 설명해주세요.
답변: ## 김치: 한국의 상징, 역사와 문화적 중요성

김치는 단순한 음식을 넘어 한국인의 삶과 문화를 반영하는 중요한 요소입니다. 한국사에서 김치는

1. 역사:

* **고대부터:** 김치의 기원은 불확실하지만, 주요 식재료인 배추, 국, 젓갈 등은 기원전부터 한
* **삼국시대:** 당나라의 역사서에 "한국인들은 배추를 젖게 하고 먹는다"라는 기록이 있으며, 이
* **조선시대:** 김치는 국민 음식으로 자리매김했습니다. 1800년대에는 조선 시대의 대표적인 김치

2. 문화적 중요성:

* **건

질문: 조선시대의 과거 제도에 대해 간단히 설명해주세요.
답변: 조선시대 과거는 개인의 학문적 능력을 평가하여 관료직에 임명하는 시험제도였습니다.

핵심적인 특징은 다음과 같습니다:

* **"문과, 이, 사, 진" 4과 시험:** 문과는 문학과 예술을 전문적으로 공부했으며, 이과는 과학과
* **기출문제 기반 학습:** 과거 시험은 전통적인 유학 서적을 바탕으로 기출문제를 중심으로 진행
* **고위 관료 출신의 기존 유학자 위주:** 과거 시험은 기존 유학자들이 주도하고, 유학 전통을 계

**과거 시험을 통과한 사람들은 '진사'로 불리며, 궁궐에서 고위 관료로서 임명되는 혜택을 누렸습니

조선시대 과거 제도는 당시 사회의 신분 체계와 학문적 전통을 반영하는 동시에, 개인

```
from langchain.document_loaders import PyPDFLoader
from langchain.text_splitter import CharacterTextSplitter
from langchain.embeddings import OpenAIEmbeddings
from langchain.vectorstores import FAISS
from langchain.chat_models import ChatOpenAI
from langchain.prompts import ChatPromptTemplate
from langchain.schema.runnable import RunnablePassthrough
from langchain.schema.output_parser import StrOutputParser

import os
import tempfile

# OpenAI API 키 설정
os.environ["OPENAI_API_KEY"] = "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"

# PDF 처리 함수
def process_pdf(pdf_file):
    with tempfile.NamedTemporaryFile(delete=False, suffix=".pdf") as temp_file:
        temp_file.write(pdf_file.read())
        temp_file_path = temp_file.name
```

```
    loader = PyPDFLoader(temp_file_path)
    documents = loader.load()
    text_splitter = CharacterTextSplitter(chunk_size=1000, chunk_overlap=200)
    texts = text_splitter.split_documents(documents)

    os.unlink(temp_file_path)

    return texts

# 벡터 저장소 생성
def create_vectorstore(texts):
    embeddings = OpenAIEmbeddings()
    vectorstore = FAISS.from_documents(texts, embeddings)
    return vectorstore

# 검색된 문서들을 하나의 문자열로 반환
def format_docs(docs):
    return 'WnWn'.join(doc.page_content for doc in docs)

# RAG 체인 설정
def setup_rag_chain(vectorstore):
    template = '''You are a helpful AI assistant. You are a financial expert helping small business
    Answer the question based only on following context:
    {context}

    Question: {question}
    '''

    prompt = ChatPromptTemplate.from_template(template)

    retriever = vectorstore.as_retriever()
    llm = ChatGroq(
    model="gemma2-9b-it",
    temperature=0.7,
    max_tokens=300,
    api_key="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"
    )
    rag_chain = (
        {'context': retriever | format_docs, 'question': RunnablePassthrough()}
        | prompt
        | llm
        | StrOutputParser()
    )

    return rag_chain

# 메인 함수
def main():
    # 파일 업로드 시뮬레이션
    pdf_file = open(input("업로드할 PDF 파일 경로를 입력하세요: "), "rb")

    texts = process_pdf(pdf_file)
    vectorstore = create_vectorstore(texts)
    rag_chain = setup_rag_chain(vectorstore)
```

```
    while True:
        query = input("질문을 입력하세요 (종료하려면 'exit' 입력): ")
        if query.lower() == 'exit':
            break
        result = rag_chain.invoke(query)
        print("답변:", result)

if __name__ == "__main__":
    main()

# 공무원에 대해 질문 예정 (소방대원과 구급대원의 차이점 등...)
```

```
업로드할 PDF 파일 경로를 입력하세요: /content/test.pdf
---------------------------------------------------------------------------
EmptyFileError                            Traceback (most recent call last)
<ipython-input-11-0fdf6412f4be> in <cell line: 84>()
     83
     84 if __name__ == "__main__":
---> 85     main()

                                 9 frames
/usr/local/lib/python3.10/dist-packages/pypdf/_reader.py in _basic_validation(self, stream)
    671             raise UnsupportedOperation("cannot read header")
    672         if header_byte == b"":
--> 673             raise EmptyFileError("Cannot read an empty file")
    674         elif header_byte != b"%PDF-":
    675             if self.strict:

EmptyFileError: Cannot read an empty file
```

다음 단계: 오류 설명