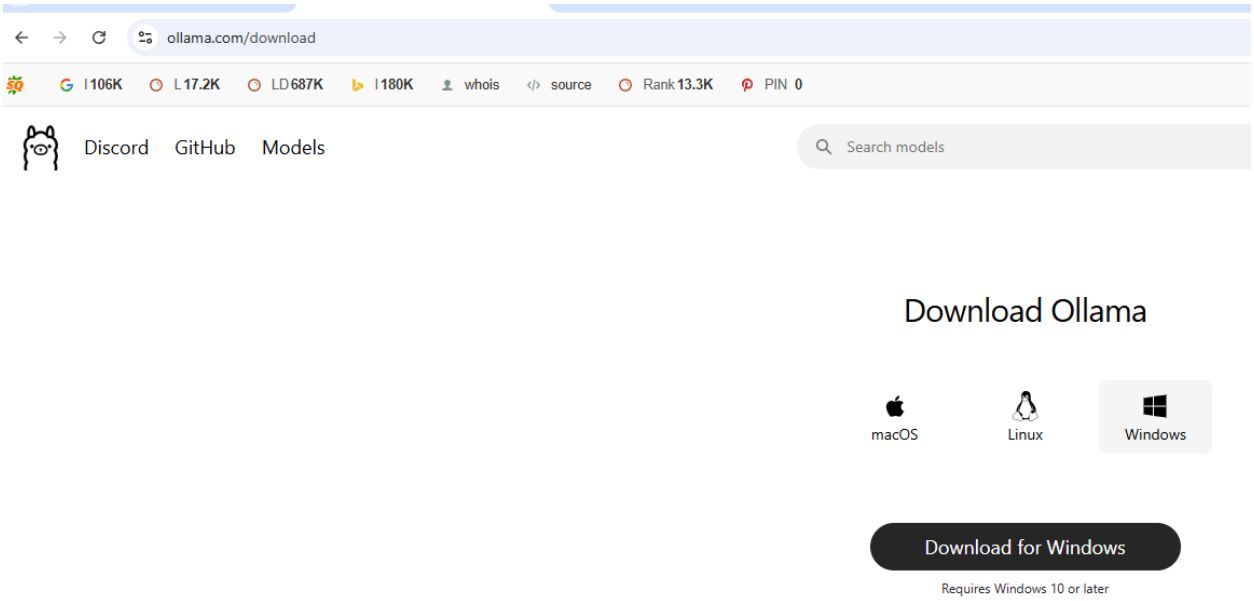
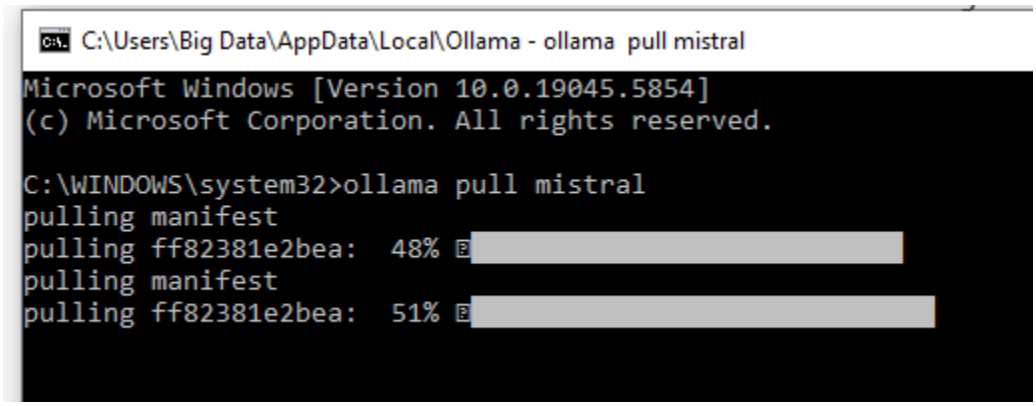
Run Mistral with Ollama (Easier for Local Devs)

1. Install Ollama: https://ollama.com



2. Run:

```
ollama pull mistral
```



3. Use with LangChain:

```
Python:-

from langchain_community.llms import Ollama

llm = Ollama(model="mistral")
```

Here's a **complete Python script** for a simple PDF Q&A agent using **Mistral 7B Instruct** via Hugging Face Transformers, plus SentenceTransformers for embeddings and FAISS for vector search.

What this script does:

- Loads a PDF and extracts text.
- Splits text into chunks.
- Embeds chunks with SentenceTransformers.
- Builds a FAISS vector store.
- Runs queries using Mistral LLM locally.
- Returns answers based on relevant PDF context.

Prerequisites:

```
pip install fitz PyMuPDF transformers sentence-transformers faiss-cpu
langchain
```

Save following python code as pdf_qa_mistral.py:-

```python
import fitz  # PyMuPDF
from transformers import AutoTokenizer, AutoModelForCausalLM, pipeline
from sentence_transformers import SentenceTransformer
import faiss
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.vectorstores import FAISS

# -------- Step 1: Extract text from PDF --------
def extract_text_from_pdf(pdf_path):
    doc = fitz.open(pdf_path)
    text = ""
    for page in doc:
        text += page.get_text()
    return text

# -------- Step 2: Split text into chunks --------
def chunk_text(text, chunk_size=1000, chunk_overlap=200):
    splitter = RecursiveCharacterTextSplitter(chunk_size=chunk_size,
chunk_overlap=chunk_overlap)
    chunks = splitter.split_text(text)
    return chunks

# -------- Step 3: Embed text chunks --------
```

```
def embed_chunks(chunks):
    embedder = SentenceTransformer("all-MiniLM-L6-v2")
    embeddings = embedder.encode(chunks)
    vectorstore = FAISS.from_texts(chunks, embeddings)
    return vectorstore

# -------- Step 4: Load Mistral model --------
def load_mistral():
    model_id = "mistralai/Mistral-7B-Instruct-v0.1"
    tokenizer = AutoTokenizer.from_pretrained(model_id)
    model = AutoModelForCausalLM.from_pretrained(model_id, device_map="auto",
torch_dtype="auto")
    generator = pipeline("text-generation", model=model, tokenizer=tokenizer)
    return generator

# -------- Step 5: Generate answer using context --------
def answer_question(generator, question, relevant_chunks):
    context = "\n\n".join(relevant_chunks)
    prompt = f"Answer the question based on the following
context:\n{context}\n\nQuestion: {question}\nAnswer:"
    output = generator(prompt, max_new_tokens=200,
temperature=0.3)[0]["generated_text"]
    # Extract answer after "Answer:"
    answer = output.split("Answer:")[-1].strip()
    return answer

# -------- Main Q&A Loop --------
def main(pdf_path):
    print("Extracting text from PDF...")
    text = extract_text_from_pdf(pdf_path)

    print("Chunking text...")
    chunks = chunk_text(text)

    print("Embedding chunks...")
    vectorstore = embed_chunks(chunks)

    print("Loading Mistral model...")
    generator = load_mistral()

    print("Ready! Ask questions about your PDF (type 'exit' to quit).")
    while True:
        query = input("Question: ").strip()
        if query.lower() in ['exit', 'quit']:
            break
```

```
        # Retrieve top 3 relevant chunks
        docs_and_scores = vectorstore.similarity_search_with_score(query, k=3)
        relevant_chunks = [doc.page_content for doc, _ in docs_and_scores]

        answer = answer_question(generator, query, relevant_chunks)
        print("\nAnswer:", answer)
        print("-" * 40)

if __name__ == "__main__":
    pdf_file = input("Enter path to your PDF file: ").strip()
    main(pdf_file)
```

1. Run:

```
python pdf_qa_mistral.py
```

3. Enter your PDF file path when prompted.
4. Ask questions about the PDF content!
5. Type `exit` to quit.