1. We have to convert the pdf file into text (because we can take operations on the text which is not possible in pdf itself).
   We have many way to do it:-
   1. By PyPDF
      Python package PyPDF can be used to achieve what we want (text extraction), although it can do more than what we need. This package can also be used to generate, decrypting and merging PDF files.

      To install: pip install PyPDF2
      Code to extract

```
# importing required modules
from PyPDF2 import PdfReader

# creating a pdf reader object
reader = PdfReader('example.pdf')

# printing number of pages in pdf file
print(len(reader.pages))

# getting a specific page from the pdf file
page = reader.pages[0]

# extracting text from page
text = page.extract_text()
print(text)
```

Now we explain each line of code

We created an object of PdfReader class from the PyPDF2 module.
The PdfReader class takes a required positional argument of the path to the pdf file.

reader = PdfReader('example.pdf')

   2. Extracting text from a PDF file using the PyMuPDF library.

To install: pip install PyMuPDF==1.16.14

Code to extract

To extract the text from the pdf, we need to follow the following steps:

1. Importing the library :- import fitz
2. Opening document :- doc = fitz.open('sample.pdf')
3. Extracting text :- pdf_text = read_pdf_mupdf(pdf_path)

   print(pdf_text)

HERE WE USE PyMupdf to extract the text from pdf.

Now,

Learn how to turn text into numbers, unlocking use cases like search.

In Python, you can split a string into tokens with OpenAI's tokenizer tiktoken.

For third-generation embedding models like text-embedding-3-small, use the cl100k_base encoding.

```
import tiktoken

def num_tokens_from_string(string: str, encoding_name: str) -> int:
    """Returns the number of tokens in a text string."""
    encoding = tiktoken.get_encoding(encoding_name)
    num_tokens = len(encoding.encode(string))
    return num_tokens


num_tokens_from_string("tiktoken is great!", "cl100k_base")
```

Embeddings

OpenAI’s text embeddings measure the relatedness of text strings. Embeddings are commonly used for:

Search (where results are ranked by relevance to a query string)
Clustering (where text strings are grouped by similarity)

Recommendations (where items with related text strings are recommended)
Anomaly detection (where outliers with little relatedness are identified)
Diversity measurement (where similarity distributions are analyzed)
Classification (where text strings are classified by their most similar label)

An embedding is a vector (list) of floating point numbers. The distance between two vectors measures their relatedness. Small distances suggest high relatedness and large distances suggest low relatedness.

Split text using tokens length using this functions.

```
def split_text_by_tokens(pdf_text,
max_tokens=4000):
    text_segments = []
    current_segment = ""

    for line in pdf_text.splitlines():
        if tokens(current_segment + line) <=
max_tokens:
            current_segment += line + "\n"
        else:
            text_segments.append(current_segment)
            current_segment = line + "\n"

    if current_segment:

text_segments.append(current_segment.strip())

    return text_segments
```

This function converts the whole text into a list of text which contains 4000 tokens.(its limit is 8172)

Now, using pandas we form a data frame of these lists using this code

.import pandas as pd

```
df = pd.DataFrame(text_segments, columns=["text"])
df.head()
```

Now we make a dataframe of text , tokens of that text and token counts

Using this code of python.

```
df['tokens'] = df['text'].apply(lambda x :
encoding.encode(x))
df['token_cnt'] = df['tokens'].apply(lambda x :
len(x))
df.head()
```

Now we have 3 column of (text, tokens,token_cnt)

Now , we create the EMBEDDING of the given tokens:

Using this codes

```
from openai import OpenAI

client = OpenAI(api_key='Kt')

def create_emb(tokens):
    return client.embeddings.create(input=tokens,
model="text-embedding-3-large").data[0].embedding
```

And add the embeddings in the given data frame.

Now we use pinecone to store the given database and work on that database

We upload the row(embedding,test, and row_number)

CONCLUSION

Our whole data of the given pdf get store in pinecone and now we do our whole operation on that database in pinecone itself.

Orgasm

1. Generate the openAi key on the openai site.https://platform.openai.com/api-keys
2. Login with pinecone and copy the api_key and index name.

This code takes text and generates the embedding (similar to previous one).

```
def create_emb(text):
    return client.embeddings.create(input=text,
model="text-embedding-3-large").data[0].embedding
```

This function takes a response containing matches and a user's query, and it creates a formatted text containing the user's query, a description line, and the relevant texts from the response.

```
def create_input(response, query):
    text = 'user\'s query :' + query + '\n'
    text += 'Following are the texts that might be
relevant the query:\n'
    for j in response['matches']:
        text+=j['metadata']['text'] + '\n'
    return text
```

This function is used to generate the answer using chatgtp.and this give answer in written way.

```
def get_answers(query):
```

```
    response = client.chat.completions.create(
  model="gpt-3.5-turbo-0125",
  messages=[
    {"role": "system", "content": "You are a
helpful assistant, who gives best reponse to the
user's query you are given the user's query along
with data that might be relevant for answering the
query. Write a response"},
    {"role": "user", "content": query}
  ]
)
    return response.choices[0].message.content
```

This takes audio and gives the result in text form.

```
query = 'audio.mp3'

query = asr(query)
emb = create_emb(query)
response = index.query(vector=emb, top_k=4,
include_values=False, include_metadata=True)

prompt = create_input(response, query)

answers = get_answers(prompt)

print(answers)
```

Similarly the next function works in such a way that it takes text in audio and gives the result in audio form.

That’s all till now.