```
!pip install openai
```

```
Collecting openai
  Downloading openai-1.9.0-py3-none-any.whl (223 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 223.4/223.4 kB 3.7 MB/s eta 0:00:00
Requirement already satisfied: anyio<5,>=3.5.0 in /usr/local/lib/python3.10/dist-packages (from openai) (3.7.1)
Requirement already satisfied: distro<2,>=1.7.0 in /usr/lib/python3/dist-packages (from openai) (1.7.0)
Collecting httpx<1,>=0.23.0 (from openai)
  Downloading httpx-0.26.0-py3-none-any.whl (75 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 75.9/75.9 kB 10.0 MB/s eta 0:00:00
Requirement already satisfied: pydantic<3,>=1.9.0 in /usr/local/lib/python3.10/dist-packages (from openai) (1.10.13)
Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-packages (from openai) (1.3.0)
Requirement already satisfied: tqdm>4 in /usr/local/lib/python3.10/dist-packages (from openai) (4.66.1)
Collecting typing-extensions<5,>=4.7 (from openai)
  Downloading typing_extensions-4.9.0-py3-none-any.whl (32 kB)
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.10/dist-packages (from anyio<5,>=3.5.0->openai) (3.6)
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio<5,>=3.5.0->openai) (1.2.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from httpx<1,>=0.23.0->openai) (2023.11.17)
Collecting httpcore==1.* (from httpx<1,>=0.23.0->openai)
  Downloading httpcore-1.0.2-py3-none-any.whl (76 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 76.9/76.9 kB 10.4 MB/s eta 0:00:00
Collecting h11<0.15,>=0.13 (from httpcore==1.*->httpx<1,>=0.23.0->openai)
  Downloading h11-0.14.0-py3-none-any.whl (58 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.3/58.3 kB 8.2 MB/s eta 0:00:00
Installing collected packages: typing-extensions, h11, httpcore, httpx, openai
  Attempting uninstall: typing-extensions
    Found existing installation: typing_extensions 4.5.0
    Uninstalling typing_extensions-4.5.0:
      Successfully uninstalled typing_extensions-4.5.0
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
llmx 0.0.15a0 requires cohere, which is not installed.
llmx 0.0.15a0 requires tiktoken, which is not installed.
tensorflow-probability 0.22.0 requires typing-extensions<4.6.0, but you have typing-extensions 4.9.0 which is incompatible.
Successfully installed h11-0.14.0 httpcore-1.0.2 httpx-0.26.0 openai-1.9.0 typing-extensions-4.9.0
```

```
!pip install pinecone-client
```

```
Collecting pinecone-client
  Downloading pinecone_client-3.0.1-py3-none-any.whl (201 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 201.0/201.0 kB 2.9 MB/s eta 0:00:00
Requirement already satisfied: certifi>=2019.11.17 in /usr/local/lib/python3.10/dist-packages (from pinecone-client) (2023.11.17)
Requirement already satisfied: tqdm>=4.64.1 in /usr/local/lib/python3.10/dist-packages (from pinecone-client) (4.66.1)
Requirement already satisfied: typing-extensions>=3.7.4 in /usr/local/lib/python3.10/dist-packages (from pinecone-client) (4.9.0)
Requirement already satisfied: urllib3>=1.26.0 in /usr/local/lib/python3.10/dist-packages (from pinecone-client) (2.0.7)
Installing collected packages: pinecone-client
Successfully installed pinecone-client-3.0.1
```

```
import openai
import pinecone
import pandas as pd
```

```
openai.api_key = 'YOUR_OPENAI_API_KEY'
pinecone.init(api_key='YOUR_PINECONE_API_KEY')


# Function to create Pinecone index and store document vectors
def create_pinecone_index(index_name, data):
    pinecone.create_index(index_name, metric="cosine")
    pinecone_index = pinecone.Index(index_name)


 # Encode text data to vectors using BERT (you may need to install transformers library)
    # Example: Use BERT to encode data
    # encoded_data = encode_text_data(data)

    # Assuming you have vectors ready, insert them into Pinecone index
pinecone_index.upsert(items=data)


# Function to retrieve relevant documents from Pinecone
def retrieve_documents(query, index_name, num_results=5):
    pinecone_index = pinecone.Index(index_name)


 # Encode the query to vector
    # query_vector = encode_text_data(query)

    # Retrieve relevant documents using Pinecone
    response = pinecone_index.query(queries=[query], top_k=num_results)

    # Extract document IDs from the response
    document_ids = response.results[0].ids

    # Fetch the documents from the database
    relevant_documents = [data[int(doc_id)] for doc_id in document_ids]

    return relevant_documents
```

```
# Function to generate an answer using OpenAI API
def generate_answer(query, documents):

# Example usage
index_name = "business_qa_index"
data = ["Document 1 content", "Document 2 content", "Document 3 content"]  # Replace with your actual data

create_pinecone_index(index_name, data)

user_query = "What is the business about?"
relevant_documents = retrieve_documents(user_query, index_name)
answer = generate_answer(user_query, relevant_documents)

print("User Query:", user_query)
print("Relevant Documents:", relevant_documents)
print("Generated Answer:", answer)


#Remember to replace placeholder values like 'YOUR_OPENAI_API_KEY' and 'YOUR_PINECONE_API_KEY' with your actual API keys.
# Also, you'll need to adapt the code based on your specific data and use case
```

Start coding or generate with AI.