```
!pip install -qU langchain-core openai supabase
```

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 438.3/438.3 kB 7.9 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 720.4/720.4 kB 26.0 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 43.6/43.6 kB 2.1 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 54.3 MB/s eta
0:00:00
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 169.9/169.9 kB 9.3 MB/s eta
0:00:00
```

```
import os
import openai
from supabase import create_client, Client
from google.colab import userdata
from langchain_core.embeddings import DeterministicFakeEmbedding

url = userdata.get("SUPABASE_URL")
key = userdata.get("SUPABASE_KEY")
api_key: str = userdata.get("OPENAI_API_KEY")
openai.api_key = api_key
supabase: Client = create_client(url, key)

embeddings = DeterministicFakeEmbedding(size=1536)

def get_context(question: str) -> str:
    # Embed the user's question
    question_embedding = embeddings.embed_query(question)

    results = []
    if "customer" in question.lower():
        query = supabase.rpc("find_related_customer",
{'question_vector': question_embedding}).execute()
    elif "product" in question.lower():
        query = supabase.rpc("find_related_products",
{'question_vector': question_embedding}).execute()
    elif "invoice" in question.lower():
        query = supabase.rpc("find_related_invoices",
{'question_vector': question_embedding}).execute()
    else:
        return "No relevant context found for the given question."

    for item in query.data:
        results.append(item)
    return results

def get_response(question: str):
    response = openai.chat.completions.create(
        model="gpt-4o-mini",
```

```
        messages=[
            {"role": "system", "content": "You are a helpful assistant
that answers questions \
                                          about the customers,
products, and invoices provided to \
                                          you in the context. Use only
the provided context to answer \
                                          questions. If the
information isn't in the context, say so."},
            {"role": "user", "content": f"Question: {question}\n\
nContext:\n{get_context(question)}"}
        ],
        max_tokens=150,
        temperature=0.7
    )
    return response.choices[0].message.content.strip()

question = "Is there a customer named Justin Chau? If so, show me his
information"
answer = get_response(question)
print("Answer:", answer)


Answer: Yes, there is a customer named Justin. Here is his
information:

- **Name**: Justin Moore
- **Email**: justin.moore@example.com
- **Balance**: 0
- **Created**: Timestamp 1738121053
- **Delinquent**: False
- **Description**: Sample customer for testing
- **Invoice Prefix**: E2A8901D
- **Next Invoice Sequence**: 2
- **Preferred Locales**: []
- **Tax Exempt**: none
- **Live Mode**: False
```