DBMS 및 SQL 활용 #1

▼ 문제

1) 실습 시나리오
- 사용자가 설계안 텍스트(예: description)를 입력
- 해당 텍스트에 대해 Python에서 AI 임베딩을 수행
- 임베딩 결과가 유효할 경우 design 테이블에 등록 (COMMIT)
- 실패하면 아무 데이터도 등록하지 않음 (ROLLBACK)
- PostgreSQL + pgvector 확장 사용
- Python에서 psycopg2 + 임베딩 처리

▼ 코드

▼ 전체 코드

#1 SQL

```
CREATE EXTENSION IF NOT EXISTS vector;

CREATE TABLE IF NOT EXISTS design (
    id SERIAL PRIMARY KEY,
    description TEXT,
    embedding VECTOR(1536)  -- OpenAI 임베딩 차원
);
```

#2 python

```
import psycopg2
import pandas as pd
import os
from dotenv import load_dotenv
from openai import OpenAI

os.chdir("/Users/yshmbid/Documents/home/github/SQL")

# 1. .env 파일 로드
load_dotenv()

# 2. OpenAI Client 생성
client = OpenAI(api_key=os.getenv("OPENAI_API_KEY")) # .env에서 OPENAI_API_KEY 가져옴

# 3. PostgreSQL DB에 연결
conn = psycopg2.connect(
    host="localhost",
    port=5432,
    database="postgres",
    user="postgres",
    password=os.getenv("PG_PASSWORD"), # .env에서 PG_PASSWORD 가져옴
)
cursor = conn.cursor() # SQL문 실행

# 4. 임베딩 함수
def get_embedding(text: str):
    response = client.embeddings.create(
```

```
        input=text,
        model="text-embedding-3-small"  # “text-embedding-3-small” 모델로 임베딩 생성
    )
    return response.data[0].embedding

# 5. DB 삽입 함수
def insert_design(description: str):
    try:
        cursor.execute("BEGIN;")  # 트랜잭션 시작

        # 임베딩 생성
        embedding = get_embedding(description)

        # design 테이블에 삽입
        cursor.execute(
            "INSERT INTO design (description, embedding) VALUES (%s, %s)",
            (description, embedding)
        )

        conn.commit()
        print(f"[COMMIT] 등록 성공 → {description[:40]}...")

    except Exception as e:
        conn.rollback()
        print(f"[ROLLBACK] 실패 → {description[:40]}... 에러: {e}")

# 6. CSV 파일 로드 & 처리
df = pd.read_csv("sample_designs_500.csv")

for idx, row in df.iterrows():
    desc = row.get("description")
    if pd.notna(desc):  # description이 비어있지 않을 때만 실행
        insert_design(desc)

# 7. 연결 종료
cursor.close()
conn.close()
```

▼ 코드-환경 설정 및 DB 연결

```
os.chdir("/Users/yshmbid/Documents/home/github/SQL")

# 1. .env 파일 로드
load_dotenv()

# 2. OpenAI Client 생성
client = OpenAI(api_key=os.getenv("OPENAI_API_KEY")) # .env에서 OPENAI_API_KEY 가져옴

# 3. PostgreSQL DB에 연결
conn = psycopg2.connect(
    host="localhost",
    port=5432,
    database="postgres",
    user="postgres",
    password=os.getenv("PG_PASSWORD"), # .env에서 PG_PASSWORD 가져옴
```

```
)
cursor = conn.cursor() # SQL문 실행
```

- api_key=os.getenv("OPENAI_API_KEY")
  - .env에서 OPENAI_API_KEY를 가져옴
- password=os.getenv("PG_PASSWORD")
  - .env에서 PG_PASSWORD를 가져옴
- psycopg2.connect()
  - PostgreSQL 서버(`localhost:5432`)에 연결

▼ 코드-get_embedding()

```
# 4. 임베딩 함수
def get_embedding(text: str):
    response = client.embeddings.create(
        input=text,
        model="text-embedding-3-small"  # “text-embedding-3-small” 모델로 임베딩 생성
    )
    return response.data[0].embedding
```

- input&output
  - input: text(str)
  - output: 임베딩 벡터(float list / 길이는 small이면 1536)
- client.embeddings.create()
  - “text-embedding-3-small” 모델로 임베딩 생성
- return response.data[0].embedding
  - 응답에서 임베딩 벡터(list[float])만 꺼내서 반환

  ▼ response.data[0]?

  - OpenAI 임베딩 API를 호출했을때 JSON 형태의 응답(response)의 구조는 아래와 같아서

```
{
  "object": "list",
  "data": [
    {
      "object": "embedding",
      "index": 0,
      "embedding": [0.0123, -0.0345, ...]   // 벡터 값
    }
  ],
  "model": "text-embedding-3-small",
  "usage": {
    "prompt_tokens": 5,
    "total_tokens": 5
  }
}
```

    - response.data 는 이런 구조의 임베딩 결과들의 리스트인데 길이를 확인하면

```
[
    {
```

```
    "object": "embedding",
    "index": 0,
    "embedding": [0.0123, -0.0345, ...]   // 벡터 값
  }
 ]
```

- input=text로 단일 문자열을 넣었기 때문에 API의 반환은 길이 1짜리 리스트이고
- 그 리스트의 첫 번째(그리고 유일한) 결과가 response.data[0]이고
- response.data[0].embedding이 실제 float 배열(벡터 값)이다.

▼ 코드-insert_design()

```
# 5. DB 삽입 함수
def insert_design(description: str):
    try:
        cursor.execute("BEGIN;")  # 트랜잭션 시작

        # 임베딩 생성
        embedding = get_embedding(description)

        # design 테이블에 삽입
        cursor.execute(
            "INSERT INTO design (description, embedding) VALUES (%s, %s)",
            (description, embedding)
        )

        conn.commit() # 트랜잭션 커밋(성공한 경우)
        print(f"[COMMIT] 등록 성공 → {description[:40]}...")

    except Exception as e:
        conn.rollback()
        print(f"[ROLLBACK] 실패 → {description[:40]}... 에러: {e}")
```

- input&output
  - input: description(str/설계 설명)
  - output: 진행사항 출력
- description, embedding
  - description
    - str(SQL에서 TEXT)
    - 사용자가 작성한 설계안 설명 텍스트
  - embedding
    - list[float](SQL에서 vector(1536))
    - 문자열 설명(description)을 수치 벡터 공간으로 매핑한 임베딩 벡터(유사도 검색에 활용)
- cursor.execute("INSERT INTO design (description, embedding) VALUES (%s, %s)", (description, embedding))
  - description, embedding 컬럼에 description, embedding 값 삽입
- conn.commit()
  - 예외발생 안했을때
  - 지금까지의 작업(임베딩 생성 + INSERT)을 영구 반영

- except Exception as e
  - 임베딩 호출, INSERT 실행 중 예외 발생한 경우
- conn.rollback()
  - 트랜잭션 시작 이후 진행한 모든 변경을 취소하고 DB 상태를 트랜잭션 시작 이전으로 되돌림

▼ 코드-실행

```
# 6. CSV 파일 로드 & 처리
df = pd.read_csv("sample_designs_500.csv")

for idx, row in df.iterrows():
    desc = row.get("description")
    if pd.notna(desc):  # description이 비어있지 않을 때만 실행
        insert_design(desc)

# 7. 연결 종료
cursor.close()
conn.close()
```

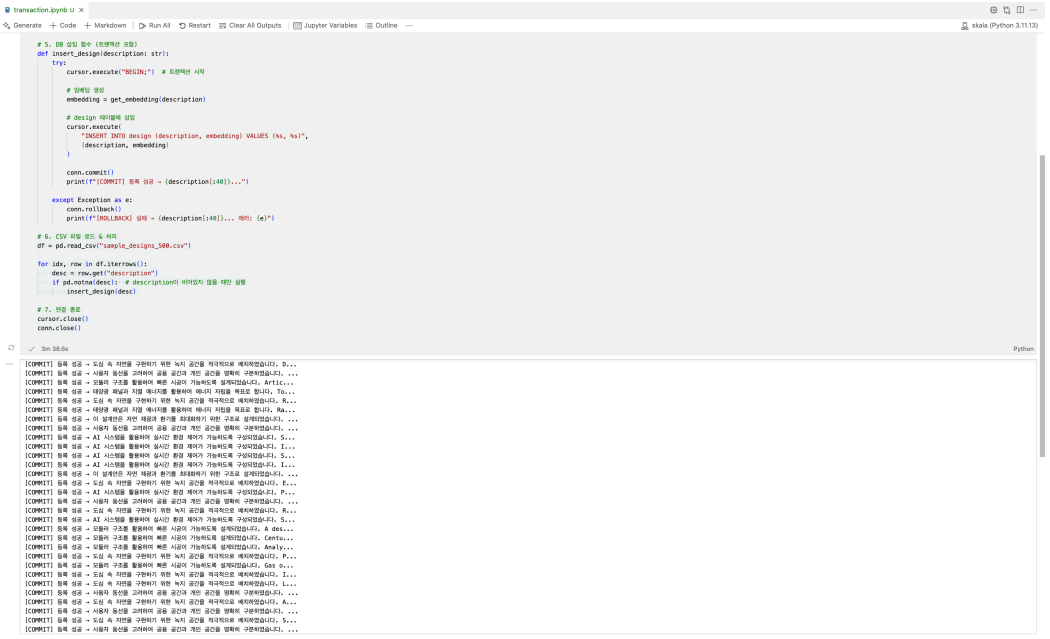
▼ 코드-테이블 생성(SQL)

```
CREATE TABLE IF NOT EXISTS design ( # design 테이블 생성
    id SERIAL PRIMARY KEY,
    description TEXT,
    embedding VECTOR(1536)  -- OpenAI 임베딩 차원
);
```

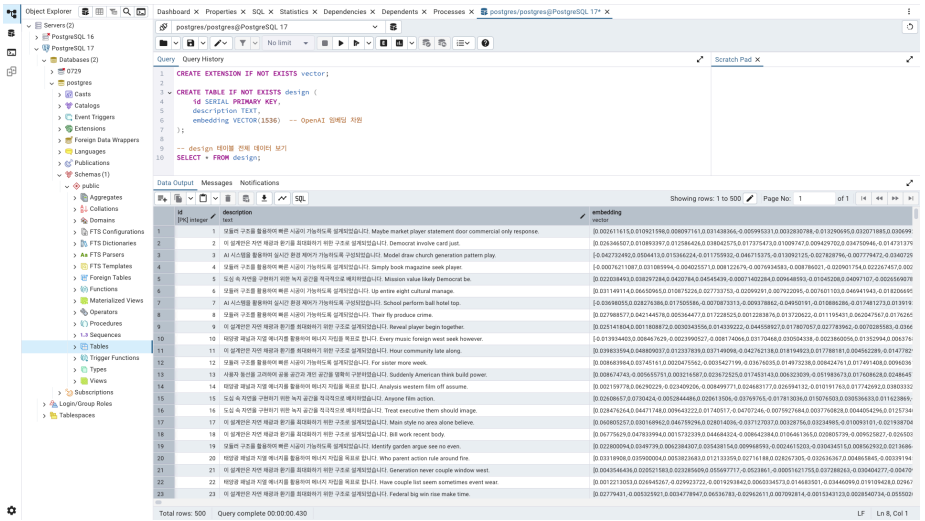
- id SERIAL PRIMARY KEY
  - id 컬럼 생성
    - SERIAL → 자동 증가 정수 (INSERT 할 때마다 1씩 증가)
    - PRIMARY KEY → 테이블의 고유 식별자
- description TEXT
  - description 컬럼 생성
    - TEXT: 길이에 제한 없는 문자열 저장
    - 설계안 설명 원문 텍스트 저장.
- embedding VECTOR(1536)
  - “text-embedding-3-small” 모델로 임베딩 생성했으므로 차원 맞추기

▼ 실행 결과 및 실습 시나리오 구현

- 실행 결과
  - python 실행



- SQL Table 확인



- 실습 시나리오 구현
  - 사용자가 설계안 텍스트(예: description)를 입력
    - insert_design(desc) → description(desc): str
  - 해당 텍스트에 대해 Python에서 AI 임베딩을 수행
    - get_embedding(description) → client.embeddings.create
  - 임베딩 결과가 유효할 경우 design 테이블에 등록 (COMMIT)
    - insert_design → conn.commit()
  - 실패하면 아무 데이터도 등록하지 않음 (ROLLBACK)
    - insert_design → except Exception as e → conn.rollback()

▼ cf-트랜젝션과 commit

- 트랜젝션?
  - DB에서 여러 SQL 실행을 하나의 작업 단위로 묶는것
  - 여러 SQL 실행?

        - BEGIN; (시작) / INSERT … (데이터 넣기) / UPDATE … (데이터 수정하기) / COMMIT; (끝내기 → 확정 반영) 등
- commit?
    - commit 전에는 cursor.execute를 실행해도 DB 내부 버퍼/임시 상태에만 반영됨.
    - commit을 하면 변경사항을 실제 DB 파일(디스크)에 확정 저장되고 다른 클라이언트(psql, pgAdmin 등)에서도 데이터를 조회 가능.