SK네트웍스 **Family AI** 과정 **12**기

데이터 수집 및 저장 데이터 조회 프로그램

산출물 단계	데이터 수집 및 저장
평가 산출물	데이터 조회 프로그램
제출 일자	2025. 08. 01
깃허브 경로	https://github.com/SKNETWORKS-FAMILY-AICAMP/SKN13-FINAL-3TEAM.git
작성 팀원	기원준, 전진혁, 강지윤, 최호연, 우민규

원본 데이터 **(**텍스트**)**	현대 모터스튜디오_디자인 관련 문서.pdf 현대 디자인 모토.txt 현대자동차 디자인 철학에 내재하는 미의식의 신경학적 해석.pdf 자동차 인테리어 설계 원리.txt 차체 및 구조 설계의 모든 것.txt car_specs.zip(.csv) new_articles.txt preview_articles.txt total_articles.txt other_issue_articles.txt other_new_articles.txt other_preview_articles.txt other_total_articles.txt interview_articles.txt hyundai_car_history.json 자동차 개발단계에서의 인간공학의 역할.pdf 자동차차체형태디자인이공기역학성능에미치는영향에대한연구.pdf 차량 개발 단계의 친환경설계 프로세스에 대한 연구.pdf issue_articles.txt preview_articles.txt 현대 그레이존 디자인.txt

데이터 전처리 과정	1. .txt 파일
	
	(1) 구조 파악: 문서 간 구분자, 필드명이 어떻게 나눠 지는지 확인
	(2) 파싱 및 분할: 각 문서를 ID 단위로 분리하고, 필요한 필드를 추출
	(3) 클렌징: 불필요한 공백, 줄바꿈, 특수기호 제거
	(4) 자연어 필드 추출: ID, Title, Contents
	(5) Json 구조 예시
	{
	"ID": "고유 문서 ID (파일명이나 생성한 ID값)",
	"Title": "문서의 제목",
	"Contents": "문서의 전체 본문 또는 필요한 내용을 문자열로 입력"
	}

	2. .pdf 파일
	
	(1) 텍스트 추출: PyMuPDF, pdfplumber 등으로 페이지별 텍스트 추출
	(2) 메타정보 수집: 표지 또는 본문에서 제목 소속/기관명 추출
	(3) 본문 구조화: 서론, 본문, 결론을 구분
	(4) 키워드 추출: TF-IDF 또는 KeyBERT로 핵심어 추출
	(5) Json 구조 예시
	{
	"title": "문서 제목",
	"organization": "소속/기관명",
	"keywords": ["키워드1", "키워드2"],
	"abstract": "문서의 요약 또는 개요를 입력합니다.",
	"main_contents": [
	{
	"section_title": "섹션 1 제목",
	"section_summary": "섹션 1의 내용을 간략히 요약"
	},
	{
	"section_title": "섹션 2 제목",
	"section_summary": "섹션 2의 내용을 간략히 요약"
	}
	],
	"conclusion": "문서의 결론이나 제안을 간략히 기술합니다.",
	"additional_notes": "필요한 경우 추가적인 주의사항이나 특이점을 기록합니다."
	}
	
	3. .csv 파일
	
	(1) CSV 구조 확인: 헤더의 명칭 및 행(row)의 개별 차량 정보 확인
	(2) 데이터 정제: 단위 제거 (mm, kg, ps, km/ℓ 등), 빈 값(연료 탱크) 처리
	(3) 키값 매핑: 한글 헤더 → 영어 키로 변환 (ex: 전장(mm) → length_mm)
	(4) Json 구조 예시
	{
	"model_name": "차량의 이름을 입력합니다.",
	"dimensions": {
	"length_mm": "차량의 전체 길이(mm)를 입력합니다.",
	"width_mm": "차량의 전체 폭(mm)을 입력합니다.",
	"height_mm": "차량의 전체 높이(mm)를 입력합니다.",
	"wheelbase_mm": "축간 거리(mm)를 입력합니다.",
	"front_tread_mm": "전륜 윤거(mm)를 입력합니다.",
	"rear_tread_mm": "후륜 윤거(mm)를 입력합니다."
	},
	"passenger_capacity": "승차 정원(명)을 입력합니다.",
	"weight_kg": "공차중량(kg)을 입력합니다."
	}

DB 사용 용도	1. Vector DB

- 문서 의미 검색, RAG의 Context
- 문서 기반의 응답, 의미 기반 유사도 검색
- 각 파일 용도 설명

구분	역할	사용 예시
Json 벡터화	컨텍스트 검색, RAG	"2023년형 넥쏘의 공기역학 성능 설명해줘" → 관련 json 찾아 응답
사용자 생성문서 (asset_library)	유저 업로드 벡터화	업로드한 문서 기반 문맥 질의, 개인화된 분석
사용자 리뷰 (user_review.mentioned_features)	임베딩 후 감성 클러스터링	특정 기능 언급된 리뷰들 의미기반 그룹핑
디자인 가이드/CI	규정 문서 기반 제안	"이 모델의 전면부 로고 위치 적절해?" → CI 문서와 비교 / 이미지 생성 및 마스킹에 반영

2. RDB
- 검색, 통계, 필터링, 사용자 데이터 저장에 활용
- VectorDB/LLM 응답의 메타조건 필터로 활용
- 테이블 설명

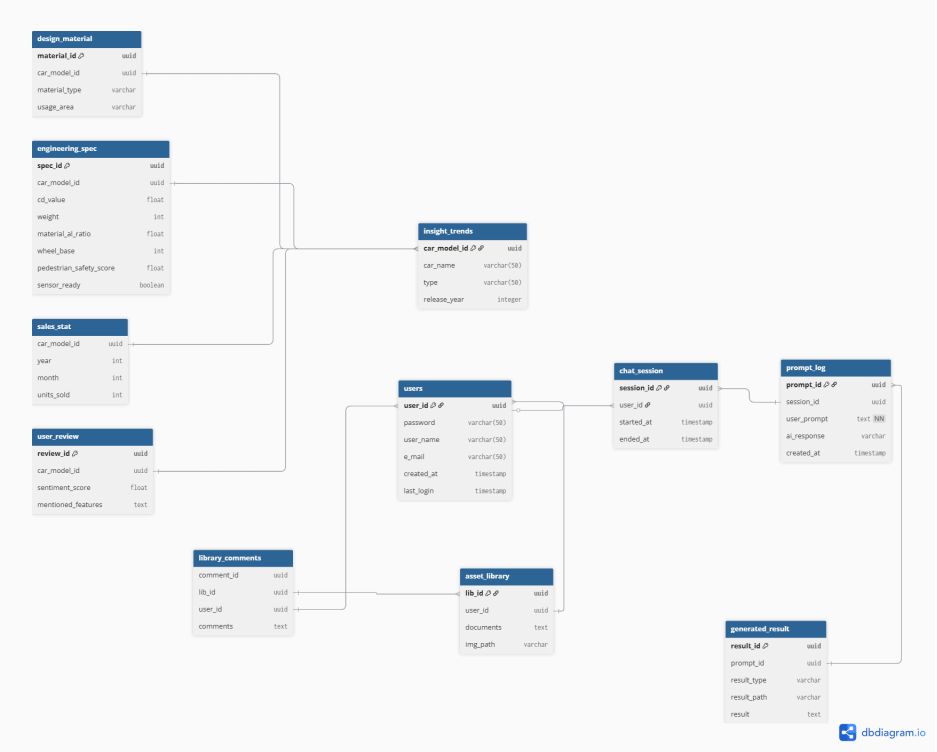
테이블	용도	설명
users	사용자 정보	사용자 계정, 이메일, 로그인 시간 등 기본 인증 정보 저장
chat_session	LLM 사용 로그	유저별 채팅 세션 관리 (session_id, 시작/종료시간 등)
prompt_log	프롬프트 기록	사용자가 입력한 프롬프트 + LLM 응답 + 생성 시간 기록
generated_result	결과물 기록	프롬프트 결과물 유형, 저장 경로 등 관리
asset_library	사용자의 문서/이미지 자산	벡터화 대상 문서 저장 및 연결

<table>
<tr><td></td><td>
<table>
<tr><td></td><td></td><td>(VectorDB와 연동 가능)</td></tr>
<tr><td>library_comments</td><td>자산에 대한 코멘트</td><td>유저가 업로드 자산에 남긴 코멘트 관리</td></tr>
<tr><td>insight_trends</td><td>차량 모델 기본정보</td><td>car_name, 타입, 연식 등 검색/통계용 기본 메타정보</td></tr>
<tr><td>engineering_spec</td><td>엔지니어링 제원</td><td>차량의 CD값, 무게, 휠베이스, 안전성 등 수치형 기술 스펙</td></tr>
<tr><td>design_material</td><td>사용된 소재</td><td>차종별 소재 종류, 용도 구분 (ex: 내장, 외장)</td></tr>
<tr><td>sales_stat</td><td>판매 통계</td><td>차종별 연도/월별 판매량 (예측/분석용 활용 가능)</td></tr>
<tr><td>user_review</td><td>감성 피드백</td><td>사용자 자동차 시승기 / 해당 차에 대한 평가</td></tr>
</table>
</td></tr>
<tr><td>VectorDB 구조</td><td>1. txt
{
"Hash": "uuid값을 입력합니다",
"Vector": "본문 임베딩 값을 입력합니다",
"Metadata": {
"ID": "고유 문서 ID (파일명이나 생성한 ID값)",
"Title": "문서의 제목",
"Contents": "문서의 전체 본문 또는 필요한 내용을 문자열로 입력"
}
}

2. pdf
{
"Hash": "uuid값을 입력합니다",
"Vector": "본문 임베딩 값을 입력합니다",
"Metadata": {
"title": "문서 제목",
"type": "document_type",
"organization": "소속/기관명",
"keywords": ["키워드1", "키워드2"],
"abstract": "문서의 요약 또는 개요를 입력합니다.",
"main_contents": [</td></tr>
</table>

```
    {
      "section_title": "섹션 1 제목",
      "section_summary": "섹션 1의 내용을 간략히 요약"
    },
    {
      "section_title": "섹션 2 제목",
      "section_summary": "섹션 2의 내용을 간략히 요약"
    }
  ],
  "conclusion": "문서의 결론이나 제안을 간략히 기술합니다.",
  "additional_notes": "필요한 경우 추가적인 주의사항이나 특이점을
기록합니다."
 }
}

3. csv
{
 "Hash": "uuid값을 입력합니다",
 "Vector": "본문 임베딩 값을 입력합니다",
 "Metadata": {
  "model_name": "차량의 이름을 입력합니다.",
  "dimensions": {
   "length_mm": "차량의 전체 길이(mm)를 입력합니다.",
   "width_mm": "차량의 전체 폭(mm)을 입력합니다.",
   "height_mm": "차량의 전체 높이(mm)를 입력합니다.",
   "wheelbase_mm": "축간 거리(mm)를 입력합니다.",
   "front_tread_mm": "전륜 윤거(mm)를 입력합니다.",
   "rear_tread_mm": "후륜 윤거(mm)를 입력합니다."
  },
  "passenger_capacity": "승차 정원(명)을 입력합니다.",
  "weight_kg": "공차중량(kg)을 입력합니다."
 }
}
```

Django RDB구조	
검증 코드 및 프롬프트 요약	-
검증 결과	-