**9조 Everytime 구현**

202135784 송민의 / 202135835 정지호 / 202139853 정용준/ 202235004 고효경 / 202334455 남윤정

1. **App & View introduction**

**- App Overview**  
: Everytime은 대학생들이 학교 생활을 효율적으로 관리할 수 있도록 설계된 모바일 애플리케이션입니다.

주요기능

* 시간표 관리: 수업 일정을 효과적으로 관리.
* 커뮤니티 소통: 사용자 간 정보 공유와 소통.
* 강의 평가: 강의에 대한 후기를 통해 학습 환경 개선에 기여.

**- View Overview**  
: Everytime 앱의 핵심 기능은 다음 \*\*세 가지 주요 뷰(View)\*\*로 구성됩니다:

* 게시판 View
* 게시물 View
* 강의 평가 View

**1. 게시판 View**  
게시판 View는 다양한 주제로 분류된 게시판 목록을 표시하며, 사용자가 게시판을 선택하고 해당 게시글을 열람할 수 있도록 합니다.

**주요 기능**:

게시판 제목 표시 / 사용자가 고정한 게시판 식별 / 게시판별 당일 게시글 유무 확인

텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진

자동 생성된 설명

**2. 게시물 View**  
게시물 View는 특정 게시판에서 선택된 게시글의 상세 정보를 제공하는 화면입니다.

**제공 정보**:

* 게시글 제목, 본문, 작성일, 작성자 정보.
* 댓글 작성 및 좋아요 기능으로 상호작용 지원.

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

**3. 강의 평가 View**  
강의 평가 View는 학생들이 강의 후기를 공유하거나 열람할 수 있는 공간입니다.

제공 정보:

* 강의명(course\_name), 교수명(professor\_name), 평점(average\_rating).
* 강의 후기(course\_evaluation) 열람 및 작성 기능.

기능 세부 사항:

* CourseInfo: 강의 기본 정보를 표시.
* CourseEvaluationInfo: 집계된 평점, 과제량, 조모임 빈도 등의 데이터를 제공.
* CourseEvaluationDetails: 개별 강의평의 상세 내용을 표시.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

1. **Database design**

**[게시판 view]**

**텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명**

**Data Needed to Compose the View**

1. 게시판 제목: 각 게시판의 이름을 표시합니다.
2. 사용자가 고정한 게시판: 고정된 게시판은 User\_fixed\_bbs 테이블의 fix=1 조건으로 식별합니다.
3. 당일 게시글 여부: 특정 날짜(예: 2024-11-20)에 게시글이 존재하는지 확인하여 표시합니다.

**Required Tables for the View**

1. **게시판** (bbs): 게시판 정보 저장.
   * bbs\_id (PK): 게시판 고유 번호.
   * bbs\_name: 게시판 이름.
2. **게시글** (contents): 게시판 내 게시글 정보를 저장.
   * contents\_id (PK): 게시글 고유 번호.
   * bbs\_id: 게시판 ID.
   * bbs\_date: 게시글 작성 날짜.
3. **유저** (user): 유저 정보를 저장.
   * i\_user (PK): 유저 고유 번호.
   * nickname: 유저 이름.
4. **유저고정** (user\_fixed\_bbs): 유저가 고정한 게시판 정보를 저장.
   * i\_user: 유저 고유 번호.
   * bbs\_id: 게시판 ID.
   * fix: 고정 여부 (1=고정, 0=비고정).
   * update\_time: 고정한 시간.

**User Fixed 게시판 View**

유저별 고정 게시판 목록을 조회하며, 고정 여부는 fix=1 조건으로 필터링합니다.

* 고정 게시판 조회: 가장 최근에 고정한 update\_time을 기준으로 fix=1 데이터를 선택.
* 결과 저장: 유저별 고정 게시판 리스트는 user\_fixed\_bbs\_top 뷰에 저장.

**SQL Query:**

*CREATE VIEW user\_fixed\_bbs\_top AS*

*SELECT ufb.i\_user, ufb.bbs\_id*

*FROM user\_fixed\_bbs ufb*

*JOIN (*

*SELECT i\_user, bbs\_id, MAX(update\_time) AS update\_time*

*FROM user\_fixed\_bbs*

*GROUP BY i\_user, bbs\_id*

*) a*

*ON ufb.i\_user = a.i\_user*

*AND ufb.bbs\_id = a.bbs\_id*

*AND ufb.update\_time = a.update\_time*

*WHERE ufb.fix = 1*

*ORDER BY ufb.i\_user, ufb.bbs\_id;*

**게시판 정보 나열하기**

목표: 유저별 게시판 목록을 출력하며 **고정 여부**, **당일 게시글 존재 여부**를 확인합니다.

조회 과정

1. 게시판 이름과 고정 여부:
   * bbs 테이블에서 게시판 제목을 가져옵니다.
   * user\_fixed\_bbs\_top에서 유저의 고정 게시판을 확인합니다.
2. 새로운 게시글 여부:
   * contents 테이블에서 해당 날짜(DATE(bbs\_date) = '2024-11-20')의 게시글 존재 여부를 확인합니다.
   * 게시글이 있으면 1, 없으면 0 표시.
3. 게시판 정렬 기준:
   * 고정된 게시판 우선 정렬.
   * 일반 게시판은 bbs\_id 기준 오름차순 정렬.

**SQL Query:**

*SELECT*

*bbs.bbs\_name,*

*IF(*

*EXISTS (*

*SELECT 1*

*FROM contents*

*WHERE contents.bbs\_id = bbs.bbs\_id AND DATE(contents.bbs\_date) = '2024-11-20'*

*),*

*1,*

*0*

*) AS has\_new\_posts*

*FROM*

*bbs*

*LEFT JOIN user\_fixed\_bbs\_top ufbt*

*ON bbs.bbs\_id = ufbt.bbs\_id AND ufbt.i\_user = 1*

*ORDER BY*

*CASE*

*WHEN bbs.bbs\_id in(1,2,3,4,5) then 0*

*WHEN ufbt.bbs\_id IS NOT NULL THEN 1 -- 고정 게시판*

*ELSE 2 -- 일반 게시판*

*END,*

*bbs.bbs\_id*

**result:**

**텍스트, 스크린샷, 번호, 폰트이(가) 표시된 사진

자동 생성된 설명**

**API Implementation**

사용자가 i\_user 값을 입력하면 해당 유저의 게시판 정보를 반환하는 API입니다.

*from flask import Flask, request, jsonify*

*import pymysql*

*import pandas as pd*

*app = Flask(\_\_name\_\_)*

*@app.route('/bbs', methods=['POST'])*

*def get\_bbs\_info():*

*request\_json = request.get\_json()*

*user\_id = request\_json['i\_user']*

*conn = pymysql.connect(*

*host='localhost',*

*port=3306,*

*user='root',*

*password='1234',*

*db='everytime'*

*)*

*sql\_query = f"""*

*SELECT*

*bbs.bbs\_name,*

*IF(*

*EXISTS (*

*SELECT 1*

*FROM contents*

*WHERE contents.bbs\_id = bbs.bbs\_id AND DATE(contents.bbs\_date) = '2024-11-20'*

*),*

*1,*

*0*

*) AS has\_new\_posts*

*FROM*

*bbs*

*LEFT JOIN user\_fixed\_bbs\_top ufbt*

*ON bbs.bbs\_id = ufbt.bbs\_id AND ufbt.i\_user = {user\_id}*

*ORDER BY*

*CASE*

*WHEN bbs.bbs\_id in (1,2,3,4,5) THEN 0*

*WHEN ufbt.bbs\_id IS NOT NULL THEN 1*

*ELSE 2*

*END,*

*bbs.bbs\_id;*

*"""*

*try:*

*df\_bbs\_info = pd.read\_sql\_query(sql\_query, conn)*

*return df\_bbs\_info.to\_json(orient="records")*

*except Exception as e:*

*return jsonify({"error": str(e)})*

*finally:*

*conn.close()*

*if \_\_name\_\_ == '\_\_main\_\_':*

*app.run(debug=True)*

**Result :**

스크린샷, 소프트웨어, 멀티미디어 소프트웨어, 텍스트이(가) 표시된 사진

자동 생성된 설명

**[게시물 view]**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

**Data Needed to Compose the View**

1. 게시물 제목 (bbs\_title): 게시물의 제목을 표시합니다.
2. 게시물 내용 (bbs\_content**)**: 게시물의 본문을 표시합니다.

50자 이상일 경우, ...으로 축약 표시.

1. 작성일시 (bbs\_date): 게시물이 작성된 시간을 표시합니다.
2. 작성일시가 최근 60초 이내일 경우: '방금'

* 60분 이내일 경우: 'X분 전'
* 오늘 작성된 게시글일 경우: 'Hour:Minutes' 형식
* 그 외의 경우: 'Month/Date' 형식

1. 닉네임 (is\_anonoymous**)**: 1이라면 작성자를 익명으로, 0이라면 닉네임으로 표시합니다.
2. 좋아요 수 (bbs\_likes): 게시물의 좋아요 수를 표시합니다. 1개 이상일 경우에만 표시됩니다.
3. 댓글 수 (bbs\_comments): 게시물에 달린 댓글 수를 표시합니다. 1개 이상일 경우에만 표시됩니다.
4. 삭제 여부 (bbs\_available): 삭제된 게시물인지 판단하는 값입니다. 값이 1인 경우만 사용자에게 보여집니다.

**Kind of Tables Needed**

Contents (게시물 테이블)

* contents\_id (PK): 게시물의 고유 번호. (자동 증가).
* bbs\_id (FK): 게시물이 속한 게시판의 ID.
* bbs\_title: 게시물의 제목.
* bbs\_content: 게시물의 본문.
* bbs\_date: 게시물 작성일시.
* i\_user: 게시물 작성자의 닉네임.
* bbs\_likes: 게시물의 좋아요 수.
* bbs\_comments: 게시물의 댓글 수.
* i\_user (FK): 게시물을 작성한 사용자의 고유 ID.
* bbs\_available: 게시물의 삭제 여부. (1: 노출, 0: 삭제)

User (사용자 테이블)

* i\_user: 사용자의 고유 번호.
* nickname: 사용자의 닉네임.

주요 속성 설명

1. **contents\_id**
   * 게시물의 고유 번호로, 자동 증가(auto\_increment) 속성이 부여
   * 작성 순서대로 부여되며, 값이 높을수록 최근에 생성된 데이터.
   * 시간순으로 게시물을 정렬할 때 bbs\_date 대신 **contents\_id**를 기준으로 정렬하는 것이 성능상 더 효율적.
2. **bbs\_content**
   * 게시물의 본문 내용은 겉보기에는 글자 수나 행 수에 따라 가공하여 가져오는 것처럼 보일 수 있지만

실제로는 전체 내용을 쿼리로 가져온 후 프론트엔드에서처리하는 방식.

* + 이를 개선하기 위해, 본문이 \*\*50자 이상일 경우 '…'\*\*을 추가하는 방식으로 제한

1. **bbs\_date**
   * 작성일시를 가공하여 사용자에게 친숙한 형식으로 제공
   * 형식:
     + 현재 시간으로부터 60초 이내: '방금'
     + 60분 이내: 'X분 전'
     + 오늘 작성됨: 'Hour:Minutes'
     + 그 외: 'Month/Date'
2. **nickname**
   * 게시물을 작성할 때 사용자는 익명 여부를 선택.
   * 이를 위해 데이터베이스에 익명 여부를 판단할 수 있는 별도 컬럼을 추가하거나, 게시물 작성 시 nickname을 '익명'으로 설정하는 방식도 가능
   * nickname이 '익명'으로 저장되더라도 실제 작성자는 i\_user 컬럼을 통해 확인할 수 있음
3. **bbs\_available**
   * 사용자가 게시물을 삭제할 때, 대부분의 서비스는 데이터를 실제 삭제하지않고 삭제 여부만 관리
   * 이를 위해 bbs\_available 컬럼이 사용되며, 기본값은 1로 설정
     + **1**: 사용자에게 게시물이 노출
     + **0**: 삭제된 상태로 사용자에게 표시 안 됨
   * 게시물이 삭제되면 bbs\_available **= 0**으로 설정되며, 관리자가 삭제한 경우에도 동일

**SQL Query:**

**Result :**

*select contents\_id, bbs\_title,*

*# 본문 글자를 최대 50글자만 가져온다. 그 이상이면 '...'으로 표시.*

*if(CHAR\_LENGTH(bbs\_content) > 50, CONCAT(LEFT(bbs\_content, 47), '...'), bbs\_content) as bbs\_content,*

*# date를 가공한다. ex) 방금, 'x분전', 23:00, 11/21*

*if(*

*curdate() = date(bbs\_date),*

*if(date\_sub(current\_timestamp(), INTERVAL 60 minute) < bbs\_date,*

*if(date\_sub(current\_timestamp(), INTERVAL 60 second) < bbs\_date,*

*'방금',*

*concat(TIMESTAMPDIFF(minute, bbs\_date, current\_timestamp()), '분 전')),*

*date\_format(bbs\_date, '%H:%i')),*

*DATE\_FORMAT(bbs\_date, '%m/%d')) as date,*

*if(is\_anonymous, "익명", u.nickname) nickname, bbs\_likes, bbs\_comments from contents c*

*join user u on c.i\_user = u.i\_user*

*where bbs\_id = 8 # 이부분에 post로 값이 들어간다.*

*and bbs\_available = 1*

*order by contents\_id desc*

*limit 20;*

텍스트, 번호, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

**API Implementation**

|  |
| --- |
| *from flask import Flask, request*  *import pymysql*  *import pandas as pd*  *app = Flask(\_\_name\_\_)*  *# contents\_query*  *@app.route('/contents\_list', methods=['POST'])*  *def contents\_query():*  *request\_json = request.get\_json()*  *bbs\_input = request\_json['bbs\_id']*  *conn = pymysql.connect(host='localhost', port=3306, user='root', password='', db='everytime')*  *sql = """*  *select contents\_id, bbs\_title,*  *# 본문 글자를 최대 50글자만 가져온다. 그 이상이면 '...'으로 표시.*  *if(CHAR\_LENGTH(bbs\_content) > 50, CONCAT(LEFT(bbs\_content, 47), '...'), bbs\_content) as bbs\_content,*  *# date를 가공한다. ex) 방금, 'x분전', 23:00, 11/21*  *if(*  *curdate() = date(bbs\_date),*  *if(date\_sub(current\_timestamp(), INTERVAL 60 minute) < bbs\_date,*  *if(date\_sub(current\_timestamp(), INTERVAL 60 second) < bbs\_date,*  *'방금',*  *concat(TIMESTAMPDIFF(minute, bbs\_date, current\_timestamp()), '분 전')),*  *date\_format(bbs\_date, '%%H:%%i')),*  *DATE\_FORMAT(bbs\_date, '%%m/%d')) as date,*  *if(is\_anonymous, "익명", u.nickname) nickname, bbs\_likes, bbs\_comments from contents c*  *join user u on c.i\_user = u.i\_user*  *where bbs\_id = 8 # 이부분에 post로 값이 들어간다.*  *and bbs\_available = 1*  *order by contents\_id desc*  *limit 20;*  *# 사실 limit을 통해 20개만 가져오지만, 실제로는 pagenumber value를 통해 select하는 범위도 달라집니다.*  *""" % bbs\_input*  *# format of placeholder is : %s; to describe %H, change to %%H*  *df = pd.read\_sql\_query(sql, conn)*  *df\_dict = {"contents\_id": df['contents\_id'].tolist(), "title": df['bbs\_title'].tolist(),*  *"content": df['bbs\_content'].tolist(), "date": df['date'].tolist(), "nickname": df['nickname'].tolist(),*  *"likes": df['bbs\_likes'].tolist(), "comments": df['bbs\_comments'].tolist()}*  *return df\_dict*  *if \_\_name\_\_ == "\_\_main\_\_":*  *app.run(debug=True)* |

**r****esult :**

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명

중략

스크린샷, 텍스트이(가) 표시된 사진

자동 생성된 설명

**[강의평가 view]**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

**Data Needed to Compose the View**

1. 강의 이름: 사용자가 조회하는 강의의 이름을 표시합니다.
2. 교수 이름: 강의를 담당하는 교수의 이름을 제공합니다.
3. 평가 데이터: 강의의 평가 정보로 평균 평점, 총 평가 수, 과제 양, 조모임 빈도, 성적 기준 등에 대한 세부 데이터를 포함합니다.
4. 강의 평가 내용: 사용자가 작성한 강의평의 세부 내용과 추천/비추천 수를 확인합니다.
5. 시험 정보: 해당 강의의 시험 유형, 시험 전략, 문제 예시, 학기, 시험 회차 등을 표시합니다.
6. 사용자 제재 정보: 사용자 신고 및 제재 내역을 확인합니다.

**Required Tables for the View**

**1.** 강의 기본 정보 테이블 (CourseInfo)

* 목적: 강의별로 기본 정보와 평균 평가를 표시하고, 강의 평가에 대한 세부 정보를 제공하며, 과제 양, 조모임 빈도, 성적 기준 등을 포함합니다..
* 기본 데이터 항목:
* 강의 이름
* 교수 이름
* 평균 평점
* 총 평가 수
* 과제 양 (많음, 보통, 적음)
* 조모임 빈도 (많음, 보통, 적음)
* 성적 기준 (너그러움, 보통, 깐깐함)

**SQL Query:**

*CREATE VIEW CourseInfo AS*

*SELECT*

*c.course\_id,*

*max(c.course\_name) course\_name,*

*max(c.professor\_name) professor\_name,*

*ROUND(AVG(ce.rating), 2) AS average\_rating, -- 소수점 2자리까지 평균 레이팅*

*COUNT(ce.evaluation\_id) AS total\_evaluations,*

*-- 과제 양*

*CASE*

*WHEN SUM(CASE WHEN ce.assignment\_amount = '많음' THEN 1 ELSE 0 END) / COUNT(ce.evaluation\_id) >= 0.8 THEN '많음'*

*WHEN SUM(CASE WHEN ce.assignment\_amount = '보통' THEN 1 ELSE 0 END) / COUNT(ce.evaluation\_id) >= 0.5 THEN '보통'*

*ELSE '적음'*

*END AS assignment\_amount,*

*-- 조모임 빈도*

*CASE*

*WHEN SUM(CASE WHEN ce.group\_meeting\_frequency = '많음' THEN 1 ELSE 0 END) / COUNT(ce.evaluation\_id) >= 0.8 THEN '많음'*

*WHEN SUM(CASE WHEN ce.group\_meeting\_frequency = '보통' THEN 1 ELSE 0 END) / COUNT(ce.evaluation\_id) >= 0.5 THEN '보통'*

*ELSE '적음'*

*END AS group\_meeting\_frequency,*

*-- 성적 기준*

*CASE*

*WHEN SUM(CASE WHEN ce.grading\_flexibility = '너그러움' THEN 1 ELSE 0 END) / COUNT(ce.evaluation\_id) >= 0.8 THEN '너그러움'*

*WHEN SUM(CASE WHEN ce.grading\_flexibility = '보통' THEN 1 ELSE 0 END) / COUNT(ce.evaluation\_id) >= 0.5 THEN '보통'*

*ELSE '깐깐함'*

*END AS grading\_flexibility*

*FROM*

*Course c*

*join CourseEvaluation ce on c.course\_id = ce.course\_id*

*WHERE*

*c.course\_id = '14466\_006' -- 특정 강의 ID를 지정*

*group by course\_id;*

*-- 강의 정보 조회*

텍스트, 스크린샷, 영수증, 폰트이(가) 표시된 사진

자동 생성된 설명

2.강의 평가 정보 테이블 (CourseEvaluationInfo)

* **목적**: 강의 평가에 대한 세부 정보를 제공하며, 과제 양, 조모임 빈도, 성적 기준 등을 포함합니다.
* **기본 데이터 항목:**
* 평균 평점
* 총 평가 수
* 과제 양 (많음, 보통, 적음)
* 조모임 빈도 (많음, 보통, 적음)
* 성적 기준 (너그러움, 보통, 깐깐함)

**SQL Query:**

*CREATE VIEW CourseEvaluationInfo AS*

*SELECT*

*course\_id,*

*ROUND(AVG(rating), 2) AS average\_rating, -- 소수점 2자리까지 평균 평점*

*COUNT(evaluation\_id) AS total\_evaluations,*

*-- 과제 양*

*CASE*

*WHEN SUM(CASE WHEN assignment\_amount = '많음' THEN 1 ELSE 0 END) / COUNT(evaluation\_id) >= 0.8 THEN '많음'*

*WHEN SUM(CASE WHEN assignment\_amount = '보통' THEN 1 ELSE 0 END) / COUNT(evaluation\_id) >= 0.5 THEN '보통'*

*ELSE '적음'*

*END AS assignment\_amount,*

*-- 조모임 빈도*

*CASE*

*WHEN SUM(CASE WHEN group\_meeting\_frequency = '많음' THEN 1 ELSE 0 END) / COUNT(evaluation\_id) >= 0.8 THEN '많음'*

*WHEN SUM(CASE WHEN group\_meeting\_frequency = '보통' THEN 1 ELSE 0 END) / COUNT(evaluation\_id) >= 0.5 THEN '보통'*

*ELSE '적음'*

*END AS group\_meeting\_frequency,*

*-- 성적 기준*

*CASE*

*WHEN SUM(CASE WHEN grading\_flexibility = '너그러움' THEN 1 ELSE 0 END) / COUNT(evaluation\_id) >= 0.8 THEN '너그러움'*

*WHEN SUM(CASE WHEN grading\_flexibility = '보통' THEN 1 ELSE 0 END) / COUNT(evaluation\_id) >= 0.5 THEN '보통'*

*ELSE '깐깐함'*

*END AS grading\_flexibility*

*FROM*

*CourseEvaluation*

*WHERE*

*course\_id = '14466\_006' -- 특정 강의 ID를 지정*

*GROUP BY*

*course\_id;*

2. 강의 평가 세부 정보 테이블 (CourseEvaluationDetails)

* 목적: 강의평에 대한 세부 정보를 제공합니다.
* 기본 데이터 항목:
  + 강의평 ID
  + 평균 평점
  + 수강한 학기
  + 강의평 내용
  + 총 추천 수

**SQL Query:**

*CREATE VIEW CourseEvaluationDetails AS*

*SELECT*

*evaluation\_id AS course\_evaluation\_id, -- 강의평 ID*

*AVG(rating) AS average\_rating, -- 평균 평점*

*semester,*

*GROUP\_CONCAT(content SEPARATOR '; ') AS course\_evaluation, -- 강의평 내용*

*SUM(recommend\_count) AS total\_recommend\_count -- 총 추천 수*

*FROM*

*CourseEvaluation*

*GROUP BY*

*evaluation\_id; -- 강의평 ID로 그룹화*

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

3. 시험 정보 테이블 (ExamInfoDetails)

* 목적: 특정 강의의 시험 정보에 대한 세부 사항을 표시합니다.
* 기본 데이터 항목:
  + 시험 정보 ID
  + 학기
  + 시험 회차
  + 문제 유형
  + 시험 전략
  + 문제 예시
  + 총 추천 수
  + 총 비추천 수

**SQL Query:**

*CREATE VIEW ExamInfoDetails AS*

*SELECT*

*GROUP\_CONCAT(DISTINCT exam\_info\_id SEPARATOR ', ') AS exam\_info\_ids, -- 시험 정보 ID*

*semester, -- 학기 정보*

*exam\_round, -- 시험 회차 (중간고사, 기말고사)*

*GROUP\_CONCAT(DISTINCT question\_type SEPARATOR ', ') AS question\_type, -- 중복 문제 유형*

*GROUP\_CONCAT(exam\_strategy SEPARATOR ', ') AS exam\_strategy, -- 시험 전략*

*GROUP\_CONCAT(question\_example SEPARATOR '; ') AS question\_example, -- 문제 예시*

*SUM(recommend\_count) AS total\_recommend\_count, -- 총 추천 수*

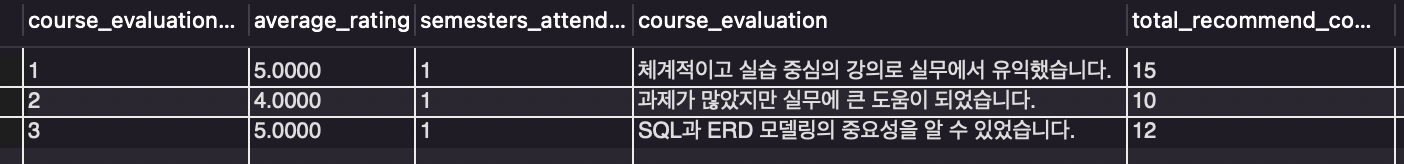
*SUM(not\_recommend\_count) AS total\_not\_recommend\_count -- 총 비추천 수*

*FROM*

*ExamInfo*

*GROUP BY*

*semester, exam\_round; -- 학기와 시험 회차로 그룹화*



4. 사용자 제재 정보 테이블 (UserSanctionDetails)

* 목적: 사용자 신고 및 제재 내역을 확인합니다.
* 기본 데이터 항목:
  + 제재 ID
  + 신고당한 사용자 ID
  + 제재 유형
  + 제재 사유
  + 제재 일시

**SQL Query:**

*CREATE VIEW UserSanctionDetails AS*

*SELECT*

*sanction\_id, -- 제재 ID*

*i\_user, -- 신고를 당한 유저 ID*

*sanction\_type, -- 제재 종류*

*reason, -- 제재 사유*

*sanction\_date -- 제재 일시*

*FROM*

*UserSanction;*

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

**API Implementation**

|  |
| --- |
| *from flask import Flask, request, jsonify*  *import pymysql*  *import pandas as pd*  *app = Flask(\_\_name\_\_)*  *# 데이터베이스 연결 설정*  *def get\_db\_connection():*  *return pymysql.connect(*  *host='localhost',*  *port=3306,*  *user='root',*  *password='',*  *db='everytime'*  *)*  *@app.route('/course\_info', methods=['POST'])*  *def get\_course\_info():*  *request\_json = request.get\_json()*  *course\_id = request\_json['course\_id']*  *sql\_query = f"""*  *SELECT \* FROM CourseInfo WHERE course\_id = '{course\_id}';*  *"""*  *try:*  *conn = get\_db\_connection()*  *df\_course\_info = pd.read\_sql\_query(sql\_query, conn)*  *except Exception as e:*  *return jsonify({"error": str(e)})*  *finally:*  *conn.close()*  *return jsonify(df\_course\_info.to\_dict(orient='records'))*  *@app.route('/course\_evaluation\_details', methods=['GET'])*  *def get\_course\_evaluation\_details():*  *sql\_query = "SELECT \* FROM CourseEvaluationDetails;"*  *try:*  *conn = get\_db\_connection()*  *df\_evaluation\_details = pd.read\_sql\_query(sql\_query, conn)*  *except Exception as e:*  *return jsonify({"error": str(e)})*  *finally:*  *conn.close()*  *return jsonify(df\_evaluation\_details.to\_dict(orient='records'))*  *@app.route('/exam\_info\_details', methods=['GET'])*  *def get\_exam\_info\_details():*  *sql\_query = "SELECT \* FROM ExamInfoDetails;"*  *try:*  *conn = get\_db\_connection()*  *df\_exam\_info\_details = pd.read\_sql\_query(sql\_query, conn)*  *except Exception as e:*  *return jsonify({"error": str(e)})*  *finally:*  *conn.close()*  *return jsonify(df\_exam\_info\_details.to\_dict(orient='records'))*  *@app.route('/user\_sanction\_details', methods=['GET'])*  *def get\_user\_sanction\_details():*  *sql\_query = "SELECT \* FROM UserSanctionDetails;"*  *try:*  *conn = get\_db\_connection()*  *df\_user\_sanction\_details = pd.read\_sql\_query(sql\_query, conn)*  *except Exception as e:*  *return jsonify({"error": str(e)})*  *finally:*  *conn.close()*  *return jsonify(df\_user\_sanction\_details.to\_dict(orient='records'))*  *if \_\_name\_\_ == "\_\_main\_\_":*  *app.run(debug=True)* |

**result :**

1.[http://127.0.0.1:5000/course\_info[POST]](http://127.0.0.1:5000/course_info*)

텍스트, 스크린샷, 소프트웨어, 웹 페이지이(가) 표시된 사진

자동 생성된 설명

2. [http://127.0.0.1:5000/course\_evaluation\_details[GET]](http://127.0.0.1:5000/course_evaluation_details*)

[텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명](http://127.0.0.1:5000/course_evaluation_details*)

3. [http://127.0.0.1:5000/exam\_info\_details[GET]](http://127.0.0.1:5000/exam_info_details%5bGET%5d)

[텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명](http://127.0.0.1:5000/exam_info_details*)

4. [http://127.0.0.1:5000/user\_sanction\_details$ [GET]](http://127.0.0.1:5000/user_sanction_details$*)

[텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명](http://127.0.0.1:5000/user_sanction_details$*)

**Tables & datasets queries view**

|  |
| --- |
| *create table bbs (bbs\_id int,*  *bbs\_name varchar(18),*  *primary key (bbs\_id));*  *INSERT INTO bbs (bbs\_id, bbs\_name) values (1, '내가 쓴 글');*  *INSERT INTO bbs (bbs\_id, bbs\_name) values (2, '댓글 단 글');*  *INSERT INTO bbs (bbs\_id, bbs\_name) values (3, '스크랩');*  *INSERT INTO bbs (bbs\_id, bbs\_name) values (4, 'HOT 게시판');*  *INSERT INTO bbs (bbs\_id, bbs\_name) values (5, 'BEST 게시판');*  *INSERT INTO bbs (bbs\_id, bbs\_name) values (6, '자유게시판');*  *INSERT INTO bbs (bbs\_id, bbs\_name) values (7, '비밀게시판');*  *INSERT INTO bbs (bbs\_id, bbs\_name) values (8, 'AI∙소프트웨어학과 정보공유');*  *INSERT INTO bbs (bbs\_id, bbs\_name) values (9, '새내기게시판');*  *INSERT INTO bbs (bbs\_id, bbs\_name) values (10, '정보게시판');*  *INSERT INTO bbs (bbs\_id, bbs\_name) values (11, '장터게시판');*  *create table user\_fixed\_bbs (i\_user int,*  *bbs\_id int, fix int, update\_time DATETIME);*  *INSERT INTO user\_fixed\_bbs (i\_user, bbs\_id, fix, update\_time) values (1, 8, 0, '2023-10-30 09:00:30');*  *INSERT INTO user\_fixed\_bbs (i\_user, bbs\_id, fix, update\_time) values (7, 6, 1, '2023-10-30 09:01:25');*  *INSERT INTO user\_fixed\_bbs (i\_user, bbs\_id, fix, update\_time) values (1, 7, 1, '2023-10-30 09:11:25');*  *INSERT INTO user\_fixed\_bbs (i\_user, bbs\_id, fix, update\_time) values (1, 8, 1, '2023-10-30 09:20:15');*  *INSERT INTO user\_fixed\_bbs (i\_user, bbs\_id, fix, update\_time) values (2, 8, 1, '2023-10-30 09:30:11');*  *INSERT INTO user\_fixed\_bbs (i\_user, bbs\_id, fix, update\_time) values (1, 6, 1, '2023-10-30 09:31:15');*  *INSERT INTO user\_fixed\_bbs (i\_user, bbs\_id, fix, update\_time) values (2, 8, 0, '2023-10-30 09:33:50');*  *INSERT INTO user\_fixed\_bbs (i\_user, bbs\_id, fix, update\_time) values (3, 6, 1, '2023-10-30 10:01:50');*  *INSERT INTO user\_fixed\_bbs (i\_user, bbs\_id, fix, update\_time) values (3, 9, 1, '2023-10-30 10:35:55');*  *INSERT INTO user\_fixed\_bbs (i\_user, bbs\_id, fix, update\_time) values (3, 7, 0, '2023-10-30 11:10:05');*  *INSERT INTO user\_fixed\_bbs (i\_user, bbs\_id, fix, update\_time) values (4, 7, 1, '2023-10-30 11:44:50');*  *INSERT INTO user\_fixed\_bbs (i\_user, bbs\_id, fix, update\_time) values (5, 10, 1, '2023-10-30 11:55:49');*  *INSERT INTO user\_fixed\_bbs (i\_user, bbs\_id, fix, update\_time) values (11, 6, 1, '2023-10-30 11:55:50');*  *INSERT INTO user\_fixed\_bbs (i\_user, bbs\_id, fix, update\_time) values (14, 7, 0, '2023-10-30 11:55:51');*  *INSERT INTO user\_fixed\_bbs (i\_user, bbs\_id, fix, update\_time) values (15, 6, 1, '2023-10-30 11:55:52');*  *create table contents(*  *contents\_id int primary key auto\_increment,*  *bbs\_id int,*  *bbs\_title varchar(40),*  *bbs\_content varchar(2048),*  *bbs\_date datetime,*  *is\_anonymous varchar(20),*  *bbs\_likes int,*  *bbs\_stars int,*  *bbs\_comments int,*  *i\_user int,*  *bbs\_available boolean default 1*  *);*  *INSERT INTO contents VALUES(NULL, 8, '캬 형 랩 미쳣다 ㅋㅋㅋㅋ', 'ㅋㅋㅋㅋㄲ', '2024-11-20 13:45', '1', 24, 1, 16, 12, 1);*  *INSERT INTO contents VALUES(NULL, 8, '앵 콜', '앵 콜', '2024-11-20 13:47', '1', 2, 0, 7, 12, 1);*  *INSERT INTO contents VALUES(NULL, 8, '드론 픽앤플레이스 잘 되나요?', '하ㅋㅋㅋㅋㅋ', '2024-11-20 16:44', '1', 0, 0, 1, 11, 0);*  *INSERT INTO contents VALUES(NULL, 8, '컴네 ta 미팅', '몇 분 정도 하나요?', '2024-11-20 20:13', '1', 0, 0, 5, 10, 1);*  *INSERT INTO contents VALUES(NULL, 8, '2학년인데 팀플때문에 죽겠어요', '데베 알고리즘 네트워크 무슨 팀플 3개를 한번에,,, 근데 기말 공부도 해야 해... 랩테도 준비해야 해... 이걸 다들 어떻게 하고 계신겁니까,,, 하루가 너무 부족해요.. 실력도 없는 것 같고 팀원들에게 폐끼치는 거 아닌가 속상해요 .. 제가 너무 작아진 것 같아요 ㅠㅠㅠ', '2024-11-20 21:12', '1', 6, 1, 13, 10, 1);*  *INSERT INTO contents VALUES(NULL, 8, '미스터 로봇 개꿀잼', '꼭봐라', '2024-11-20 21:57', '1', 0, 0, 0, 7, 1);*  *INSERT INTO contents VALUES(NULL, 8, '취창업 과제', '취창업 발표영상 올리는 과제 필수 아닌 거 맞나요?', '2024-11-20 21:57', '1', 0, 0, 0, 8, 1);*  *INSERT INTO contents VALUES(NULL, 8, '내일 드론 수업 때 완성 못 하면', '어떡하지...', '2024-11-20 23:07', '1', 0, 0, 0, 6, 1);*  *INSERT INTO contents VALUES(NULL, 8, '다죽자', '다죽자~~~ 일단 난 죽겠음', '2024-11-21 00:14', '1', 0, 0, 1, 5, 1);*  *INSERT INTO contents VALUES(NULL, 8, '안녕하세요 내년에 소웨 2학년 군복학생입니다', '하고싶은 프로젝트하면서 프레임워크 다뤄보면서 이것저것 하고있는데요, 하면서 느끼는게 전공지식이 진짜 중요하겠구나 입니다...혹시 2 3학년때 배우는 중요한 cs지식들 다 c기반으로 할까요...! 핑프라 죄송함돠!!!!', '2024-11-21 02:25', '1', 0, 0, 7, 4, 1);*  *INSERT INTO contents VALUES(NULL, 12, 'ㅇㅅㅎ 웹db', '전범위인지 중간 이후부터인지 아는사람...', '2024-11-21 03:07', '1', 0, 0, 0, 6, 1);*  *INSERT INTO contents VALUES(NULL, 12, 'ㅅㅅㅈ 자바 과제', '이거 jsp파일들이 어느디렉토리에 있는거야?', '2024-11-21 09:13', '1', 0, 0, 0, 5, 1);*  *INSERT INTO contents VALUES(NULL, 12, '코딩애플 유료수강 해보신분들 계신가요', '어떤가요 후기 알려주세요!', '2024-11-21 09:34', '1', 0, 0, 3, 9, 1);*  *INSERT INTO contents VALUES(NULL, 8, '컴구 점수 떴다 rr', '분포 레전드 ㅋㅋ', '2024-11-21 13:36', '1', 0, 0, 4, 3, 1);*  *INSERT INTO contents VALUES(NULL, 8, '알고 점수떴네', '다들 잘봄?? 평균 몇이려나', '2024-11-21 17:31', '1', 0, 0, 0, 2, 1);*  *INSERT INTO contents VALUES(NULL, 12, 'ㅇㅂㅎ 구현 다 못한사람있음?', '??', '2024-11-21 17:59', '1', 0, 0, 5, 15, 1);*  *INSERT INTO contents VALUES(NULL, 8, '슬슬', '다들 연구실 알아보는거 같은데.. 나도 알아봐야겠다', '2024-11-21 20:50', '1', 0, 0, 7, 1, 1);*  *INSERT INTO contents VALUES(NULL, 12, '카엔프', '방학에 기업나가서 실습하는거 필수인가요 선배님들', '2024-11-21 20:51', '1', 0, 0, 1, 14, 1);*  *INSERT INTO contents VALUES(NULL, 12, 'p프', '다음주부터 시작인데 논술보는데 아무런 공지가없네ㅜ', '2024-11-22 00:52', '1', 0, 0, 0, 13, 1);*  *INSERT INTO contents VALUES(NULL, 6, '대학원 권유', '어떤식으로 받음? 교수님하고 부딪힐 일이 있나? 그냥 수업하고 땡 수업하고 땡 시험보고 땡 이런식인데? 시험을 잘봐서 제안이 오는거야?', '2024-11-22 19:22', '0', 0, 0, 0, 9, 1);*  *CREATE TABLE Course (*  *course\_id VARCHAR(20) PRIMARY KEY,*  *course\_name VARCHAR(100),*  *professor\_name VARCHAR(100),*  *textbook\_info TEXT,*  *exam\_info TEXT*  *);*  *INSERT INTO Course (course\_id, course\_name, professor\_name, textbook\_info, exam\_info) VALUES ('DB101', '데이터베이스 및 실습', '조풍진 교수님', '데이터베이스 시스템, 5판 (이성환 역)', '기말고사: ER 모델링 및 SQL 실습');*  *CREATE TABLE CourseEvaluation (*  *evaluation\_id INT PRIMARY KEY,*  *course\_id VARCHAR(20),*  *evaluator\_id INT,*  *semester VARCHAR(20),*  *rating INT CHECK (rating BETWEEN 1 AND 5),*  *content TEXT,*  *recommend\_count INT, -- 추천 수*  *not\_recommend\_count INT, -- 비추천 수*  *assignment\_amount ENUM('많음', '보통', '없음'), -- 과제 양*  *group\_meeting\_frequency ENUM('많음', '보통', '없음'), -- 조모임 빈도*  *grading\_flexibility ENUM('너그러움', '보통', '없음'), -- 성적 기준*  *FOREIGN KEY (course\_id) REFERENCES Course(course\_id)*  *);*  *INSERT INTO CourseEvaluation (evaluation\_id, course\_id, evaluator\_id, semester, rating, content, recommend\_count, not\_recommend\_count, assignment\_amount, group\_meeting\_frequency, grading\_flexibility)*  *VALUES (1, '14466\_006', 202201, '2022-1학기', 5, '체계적이고 실습 중심의 강의로 실무에서 유익했습니다.', 15, 1, '많음', '보통', '너그러움');*  *INSERT INTO CourseEvaluation (evaluation\_id, course\_id, evaluator\_id, semester, rating, content, recommend\_count, not\_recommend\_count, assignment\_amount, group\_meeting\_frequency, grading\_flexibility)*  *VALUES (2, '14466\_006', 202202, '2022-2학기', 4, '과제가 많았지만 실무에 큰 도움이 되었습니다.', 10, 2, '많음', '보통', '보통');*  *INSERT INTO CourseEvaluation (evaluation\_id, course\_id, evaluator\_id, semester, rating, content, recommend\_count, not\_recommend\_count, assignment\_amount, group\_meeting\_frequency, grading\_flexibility)*  *VALUES (3, '14466\_006', 202301, '2023-1학기', 5, 'SQL과 ERD 모델링의 중요성을 알 수 있었습니다.', 12, 0, '보통', '많음', '너그러움');*  *INSERT INTO CourseEvaluation (evaluation\_id, course\_id, evaluator\_id, semester, rating, content, recommend\_count, not\_recommend\_count, assignment\_amount, group\_meeting\_frequency, grading\_flexibility)*  *VALUES (4, '14466\_006', 202302, '2023-2학기', 3, '수업 내용은 좋았지만 과제 난이도가 너무 높았습니다.', 8, 3, '많음', '보통', '보통');*  *INSERT INTO CourseEvaluation (evaluation\_id, course\_id, evaluator\_id, semester, rating, content, recommend\_count, not\_recommend\_count, assignment\_amount, group\_meeting\_frequency, grading\_flexibility)*  *VALUES (5, '14466\_006', 202401, '2024-1학기', 4, '온라인 강의로 진행되었으나 자료 준비가 철저했습니다.', 9, 1, '보통', '보통', '너그러움');*  *CREATE TABLE ExamInfo (*  *exam\_info\_id INT PRIMARY KEY, -- 시험 정보의 고유 ID*  *course\_id VARCHAR(20), -- 관련 강의의 ID*  *creator\_id INT, -- 시험을 만든 사람의 ID*  *semester VARCHAR(20), -- 학기 (예: '2024-1학기', '2024-2학기')*  *exam\_round VARCHAR(20), -- 시험 회차 (예: '중간고사', '기말고사')*  *exam\_strategy TEXT, -- 시험 전략*  *question\_type TEXT, -- 문제 유형 (중복 가능)*  *question\_example TEXT, -- 문제 예시 (2가지 이상)*  *recommend\_count INT, -- 추천 수*  *not\_recommend\_count INT, -- 비추천 수*  *FOREIGN KEY (course\_id) REFERENCES Course(course\_id)*  *);*  *INSERT INTO ExamInfo (exam\_info\_id, course\_id, creator\_id, semester, exam\_round, exam\_strategy, question\_type, question\_example, recommend\_count, not\_recommend\_count)*  *VALUES*  *(1, '14466\_006', 301, '2024-1학기', '중간고사', 'ER 모델링과 SQL 문제 위주', '객관식, 주관식', 'ER 모델링 문제, SQL 질의 작성 문제', 20, 0),*  *(2, '14466\_006', 302, '2024-1학기', '기말고사', '데이터베이스 설계 중심', '주관식, 약술형', '정규화 과정 설명, SQL 최적화 방법', 15, 1),*  *(3, '14466\_006', 303, '2024-2학기', '중간고사', '이론 및 실습 병행', '객관식, T/F형', 'DBMS의 특징, 트랜잭션의 원자성', 18, 2),*  *(4, '14466\_006', 304, '2024-2학기', '기말고사', '과제 기반 평가', '논술형, 구술', '실습 프로젝트 설명, 데이터베이스 설계 문제', 12, 1);*  *CREATE TABLE user (*  *i\_user INT PRIMARY KEY, -- 유저의 고유 ID*  *student\_id INT, -- 학번*  *user\_nm VARCHAR(100), -- 유저 이름*  *user\_id VARCHAR(50) UNIQUE, -- 유저 로그인 ID*  *user\_pw VARCHAR(100), -- 유저 비밀번호*  *ph VARCHAR(20), -- 전화번호*  *school\_id VARCHAR(20), -- 학교 ID*  *mail VARCHAR(100), -- 이메일*  *nickname VARCHAR(50) -- 닉네임*  *);*  *INSERT INTO user (i\_user, student\_id, user\_nm, user\_id, user\_pw, ph, school\_id, mail, nickname)*  *VALUES (1, 202135784, '송민의', 'user\_1', 'pass\_4392', '1088975345', 'd001', 'user1@naver.com', '민호');*  *INSERT INTO user (i\_user, student\_id, user\_nm, user\_id, user\_pw, ph, school\_id, mail, nickname)*  *VALUES (2, 202135835, '정지호', 'user\_2', 'pass\_4052', '1005680024', 'd001', 'user2@naver.com', '호용');*  *INSERT INTO user (i\_user, student\_id, user\_nm, user\_id, user\_pw, ph, school\_id, mail, nickname)*  *VALUES (3, 202135784, '정용준', 'user\_3', 'pass\_7103', '1082986134', 'd001', 'user3@naver.com', '용효');*  *INSERT INTO user (i\_user, student\_id, user\_nm, user\_id, user\_pw, ph, school\_id, mail, nickname)*  *VALUES (4, 202235004, '고효경', 'user\_4', 'pass\_5013', '1034551855', 'd001', 'user4@naver.com', '효정');*  *INSERT INTO user (i\_user, student\_id, user\_nm, user\_id, user\_pw, ph, school\_id, mail, nickname)*  *VALUES (5, 202334455, '남윤정', 'user\_5', 'pass\_7432', '1027656201', 'd001', 'user5@gmail.com', '정성');*  *INSERT INTO user (i\_user, student\_id, user\_nm, user\_id, user\_pw, ph, school\_id, mail, nickname)*  *VALUES (6, 202301766, '강민성', 'user\_6', 'pass\_4870', '1088471247', 'd002', 'user6@daum.net', '성연');*  *INSERT INTO user (i\_user, student\_id, user\_nm, user\_id, user\_pw, ph, school\_id, mail, nickname)*  *VALUES (7, 202202177, '이서연', 'user\_7', 'pass\_5774', '1052287039', 'd001', 'user7@naver.com', '연진');*  *INSERT INTO user (i\_user, student\_id, user\_nm, user\_id, user\_pw, ph, school\_id, mail, nickname)*  *VALUES (8, 202267403, '최유진', 'user\_8', 'pass\_6802', '1021533231', 'd003', 'user8@naver.com', '진민');*  *INSERT INTO user (i\_user, student\_id, user\_nm, user\_id, user\_pw, ph, school\_id, mail, nickname)*  *VALUES (9, 202380812, '오지민', 'user\_9', 'pass\_4155', '1019582741', 'd003', 'user9@gmail.com', '민우');*  *INSERT INTO user (i\_user, student\_id, user\_nm, user\_id, user\_pw, ph, school\_id, mail, nickname)*  *VALUES (10, 202395740, '윤시우', 'user\_10', 'pass\_8536', '1016061661', 'd00', 'user10@daum.net', '우강');*  *INSERT INTO user (i\_user, student\_id, user\_nm, user\_id, user\_pw, ph, school\_id, mail, nickname)*  *VALUES (11, 202193136, '강민성', 'user\_11', 'pass\_1982', '1028456768', 'd001', 'user11@daum.net', '강윤');*  *INSERT INTO user (i\_user, student\_id, user\_nm, user\_id, user\_pw, ph, school\_id, mail, nickname)*  *VALUES (12, 202300067, '정서윤', 'user\_12', 'pass\_9357', '1050358211', 'd002', 'user12@naver.com', '윤하');*  *INSERT INTO user (i\_user, student\_id, user\_nm, user\_id, user\_pw, ph, school\_id, mail, nickname)*  *VALUES (13, 202194905, '박하준', 'user\_13', 'pass\_5984', '1050386709', 'd004', 'user13@gmail.com', '하도');*  *INSERT INTO user (i\_user, student\_id, user\_nm, user\_id, user\_pw, ph, school\_id, mail, nickname)*  *VALUES (14, 202158365, '이도연', 'user\_14', 'pass\_9032', '1080882188', 'd001', 'user14@gmail.com', '도지');*  *INSERT INTO user (i\_user, student\_id, user\_nm, user\_id, user\_pw, ph, school\_id, mail, nickname)*  *VALUES (15, 202376406, '최민지', 'user\_15', 'pass\_4782', '1090740631', 'd001', 'user15@naver.com', '지민');*  *CREATE TABLE UserSanction (*  *sanction\_id INT PRIMARY KEY AUTO\_INCREMENT,*  *evaluator\_id INT, -- 신고를 당한 유저의 ID*  *sanction\_type VARCHAR(50), -- 제재 종류 (경고, 차단 등)*  *reason TEXT, -- 제재 사유*  *sanction\_date DATETIME DEFAULT CURRENT\_TIMESTAMP, -- 제재 일시*  *FOREIGN KEY (evaluator\_id) REFERENCES user(i\_user) -- 유저 정보와 연결*  *);*  *INSERT INTO UserSanction (evaluator\_id, sanction\_type, reason, sanction\_date)*  *VALUES*  *(1, '경고', '부적절한 언행', '2024-11-01 10:30:00'),*  *(2, '차단', '지속적인 스팸', '2024-11-05 14:45:00'),*  *(3, '경고', '규칙 위반', '2024-11-10 09:15:00'),*  *(4, '차단', '모욕적인 언행', '2024-11-15 11:00:00'),*  *(5, '경고', '비속어 사용', '2024-11-20 13:20:00');*  *CREATE TABLE school (*  *school\_id VARCHAR(10) PRIMARY KEY,*  *name VARCHAR(50)*  *);*  *INSERT INTO school (school\_id, name) VALUES*  *('d001', '가천대학교'),*  *('d002', '서울대학교'),*  *('d003', '연세대학교'),*  *('d004', '고려대학교');* |

**ERD models**

**텍스트, 스크린샷, 평행, 폰트이(가) 표시된 사진

자동 생성된 설명**