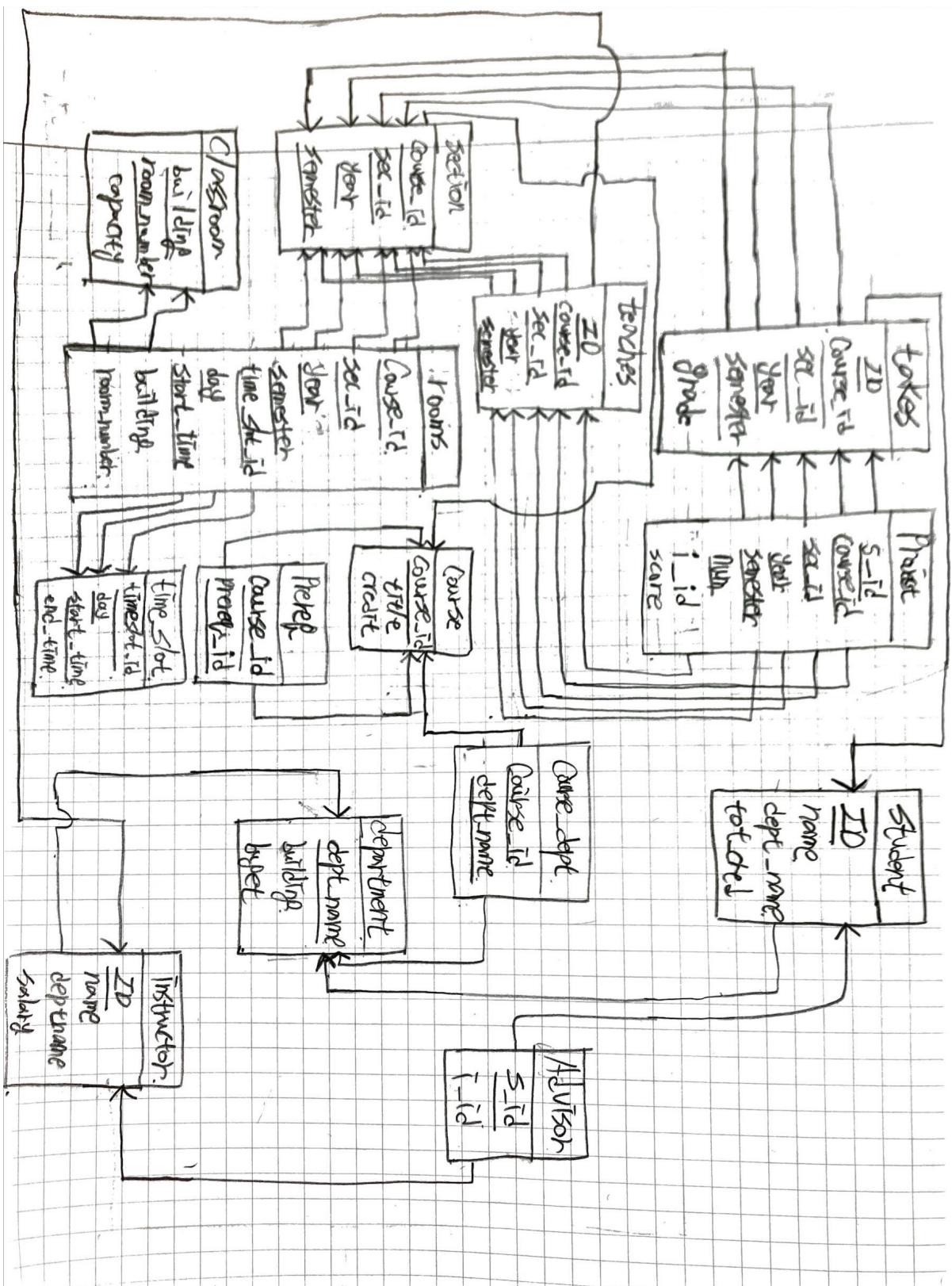




Database weak 13-14 assignment¹

20203361 장민석

1. 12주차 과제의 최종 결과로 제출한 RDB 스키마 다이어그램



2. create table문

다음과 같이 총 10개의 사용할 테이블을 생성하였다.

```
DROP TABLE IF EXISTS project, takes, student, instructor, course_dept, section, rooms,
course, time_slot;

-- Course table
CREATE TABLE course (
    course_id VARCHAR(8),
    title VARCHAR(50),
    credits NUMERIC(2,0),
    PRIMARY KEY (course_id)
);

-- Section table
CREATE TABLE section (
    course_id VARCHAR(8),
    sec_id VARCHAR(8),
    semester VARCHAR(6) CHECK (semester IN ('Fall', 'Winter', 'Spring', 'Summer')),
    year NUMERIC(4,0),
    PRIMARY KEY (course_id, sec_id, semester, year),
    FOREIGN KEY (course_id) REFERENCES course(course_id)
        ON DELETE CASCADE
);

-- Course_dept table
CREATE TABLE course_dept (
    course_id VARCHAR(8),
    dept_name VARCHAR(50),
    PRIMARY KEY (course_id, dept_name),
    FOREIGN KEY (course_id) REFERENCES course(course_id)
);

-- Time_slot table
CREATE TABLE time_slot (
    time_slot_id VARCHAR(4),
    day VARCHAR(3),
    start_time NUMERIC(2) CHECK (start_time >= 0 AND start_time < 24),
    end_time NUMERIC(2) CHECK (end_time >= 0 AND end_time < 24),
    PRIMARY KEY (time_slot_id, day, start_time)
);

-- Rooms table
CREATE TABLE rooms (
    course_id VARCHAR(8),
    sec_id VARCHAR(8),
    year NUMERIC(4,0),
    semester VARCHAR(6),
    time_slot_id VARCHAR(4),
    day VARCHAR(3),
    start_time NUMERIC(2),
    building VARCHAR(15),
    room_number VARCHAR(7),
```

```

    FOREIGN KEY (course_id, sec_id, semester, year) REFERENCES section(course_id, sec_id,
semester, year),
    FOREIGN KEY (time_slot_id, day, start_time) REFERENCES time_slot(time_slot_id, day,
start_time)
);

-- Student table
CREATE TABLE student (
    ID VARCHAR(5),
    name VARCHAR(20) NOT NULL,
    dept_name VARCHAR(50),
    tot_cred NUMERIC(3,0),
    PRIMARY KEY (ID)
);

-- Instructor table
CREATE TABLE instructor (
    ID VARCHAR(5),
    name VARCHAR(20),
    dept_name VARCHAR(50),
    salary NUMERIC(8,2),
    PRIMARY KEY (ID)
);

-- Takes table
CREATE TABLE takes (
    ID VARCHAR(5),
    course_id VARCHAR(8),
    sec_id VARCHAR(8),
    semester VARCHAR(6),
    year NUMERIC(4,0),
    grade VARCHAR(2),
    PRIMARY KEY (ID, course_id, sec_id, semester, year),
    FOREIGN KEY (course_id, sec_id, semester, year) REFERENCES section(course_id, sec_id,
semester, year),
    FOREIGN KEY (ID) REFERENCES student(ID)
);

-- Project table
CREATE TABLE project (
    s_id VARCHAR(5),
    course_id VARCHAR(8),
    sec_id VARCHAR(8),
    semester VARCHAR(6),
    year NUMERIC(4,0),
    num NUMERIC(2,0),
    name VARCHAR(50),
    i_id VARCHAR(5),
    max_score NUMERIC(3,0),
    score NUMERIC(3,0),
    PRIMARY KEY (s_id, course_id, sec_id, semester, year, num),
    FOREIGN KEY (s_id, course_id, sec_id, semester, year) REFERENCES takes(ID, course_id,
sec_id, semester, year)
);

```

3. SQL문

첫 번째 기능을 구현하기 위해 사용되는 SQL 구문은 다음과 같이 2개이다. 먼저 query1은 'course' 테이블에서 course_id, title, credits를 조회한다. 하지만 모든 강의를 조회하는 것이 아닌, 'section' 테이블과 'course_dept' 테이블에 있는 강의 중에서 주어진 year와 semester에 해당하는 강의들 중, 두 개 이상의 학과에서 개설한 과목, 즉 융합과목 만을 조회한다. 이를 위해 group by 구문과 having 구문을 사용했다. query2는 'course_dept' 테이블에서 query1에서 결과로 조회된 course_id에 해당하는 튜플을 검색한다. 이를 통해 융합과목의 dept_name 리스트를 모두 찾을 수 있다.

```
String query1 = """
    SELECT course_id, title, credits
    FROM course
    WHERE course_id IN (
        SELECT S.course_id
        FROM section as S, course_dept as CD
        WHERE (S.year, S.semester) = (?, ?)
        AND S.course_id = CD.course_id
        GROUP BY S.course_id
        HAVING COUNT(*) >= 2
    )""";
String query2 = """
    SELECT dept_name
    FROM course_dept
    WHERE course_id = ?
""";
```

두 번째 기능을 구현하기 위해 사용되는 SQL 구문은 다음과 같이 2개이다. query1은 'section' 테이블과 'course' 테이블을 조인하여, 특정 연도와 학기에 개설되는 모든 section을 찾고 이의 course_id, title, sec_id를 조회한다. query2는 위에서 찾은 section의 정보를 바탕으로 'rooms' 테이블과 'time_slot' 테이블을 조인하여, 해당 section의 요일 별 강의실의 위치 정보인 building, room_number와 수업 시간 정보인 day, start_time, end_time 정보를 찾는다.

```
String query1 = """
    SELECT S.course_id, C.title, S.sec_id
    FROM section as S, course as C
    WHERE (S.year, S.semester) = (?, ?)
    AND S.course_id = C.course_id
""";
String query2 = """
    SELECT T.day, T.start_time, T.end_time, R.building, R.room_number
    FROM rooms as R, time_slot as T
    WHERE (R.time_slot_id, R.day, R.start_time) = (T.time_slot_id, T.day,
T.start_time)
    AND (R.course_id, R.sec_id, R.year, R.semester) = (?, ?, ?, ?)
""";
```

세 번째 기능을 구현하기 위해서 총 4개의 쿼리를 실행한다. query1은 'student' 테이블에서 특정 학생 ID에 해당하는 학생의 ID, name, dept_name을 조회한다. query2는 'takes' 테이블과 'course' 테이블을 조인하여, 특정 학생이 특정 year와 semester에 수강한 section의 course_id, title, sec_id, 그리고 grade를 조회한다. query3은 query2의 결과를 바탕으로 'project' 테이블과 'instructor' 테이블을 조인하여, 특정 학생(s_id)이 들은 section(course_id, sec_id, year, semester)에서 수행한 프로젝트의 정보인 num, name, max_score, I.name, 그리고 score를 조회한다. query4는 'project' 테이블에서 집계함수를 통해 query2의 결과인 각 section에 대한 프로젝트의 총 개수(count)와 평균 점수(avg)를 계산한 결과를 검색한다.

```
String query1 = """
    SELECT ID, name, dept_name
    FROM student
    WHERE ID = ?
    """;
String query2 = """
    SELECT T.course_id, C.title, T.sec_id, T.grade
    FROM takes as T, course as C
    WHERE T.course_id = C.course_id
    AND (T.ID, T.year, T.semester) = (?, ?, ?)
    """;
String query3 = """
    SELECT P.num, P.name, P.max_score, I.name, P.score
    FROM project as P, instructor as I
    WHERE P.i_id = I.id
    AND (P.s_id, P.course_id, P.sec_id, P.year, P.semester) = (?, ?, ?, ?, ?, ?)
    """;
String query4 = """
    SELECT count(*) as count, avg(score) as avg
    FROM project
    WHERE (s_id, course_id, sec_id, year, semester) = (?, ?, ?, ?, ?, ?)
    """;
```

4. 프로그램의 로직

프로그램의 메인 함수는 다음과 같다. 우선 데이터베이스에 연결한 후 사용자로부터 입력을 받아 다양한 조회 기능을 수행한다.

```
static final String DB_URL = "----";
static final String USER = "root";
static final String PASS = "1234";
static Connection conn;
static final Scanner sc = new Scanner(System.in);

public static void main(String[] args) {
    try {
        conn = DriverManager.getConnection(DB_URL, USER, PASS);
        int command;

        while (true) {
            System.out.print("조회(1:융합과목, 2.수업 시간 및 강의실, 3.프로젝트 수행결과,
4.종료): ");
            command = sc.nextInt();
            if (command == 1)
                searchMixCourse();
            else if (command == 2)
                sectionTimeAndClassroom();
            else if (command == 3)
                searchProject();
            else if (command == 4)
                break;
            System.out.println("\n");
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        if (conn != null) {
            try {
                conn.close();
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        }
    }
}
```

다음은 융합과목을 조회하는 첫 번째 기능을 구현한 함수의 핵심 로직이다. 앞서 설명한 2 개의 쿼리에 대한 pstmt를 준비하고, pstmt1의 값은 사용자가 입력한 연도와 학기 값을 이용하여 세팅한다. 이후 반복문을 돌며 쿼리의 결과를 출력하고, 이중 반복문을 이용하여 첫 쿼리의 결과로 세팅된 pstmt2를 리스트로 출력한다.

```
PreparedStatement pstmt1 = conn.prepareStatement(query1);
PreparedStatement pstmt2 = conn.prepareStatement(query2);
```

```

System.out.print("연도 학기: ");
int year = sc.nextInt();
String semester = sc.next();

// WHERE (S.year, S.semester) = (?, ?)
stmt1.setInt(1, year);
stmt1.setString(2, semester);

try (ResultSet rs1 = stmt1.executeQuery()) {
    while (rs1.next()) {
        // SELECT course_id, title, credits
        String course_id = rs1.getString("course_id");
        System.out.print("course id: " + course_id);
        System.out.print(" | title: " + rs1.getString("title"));
        System.out.println(" | credits: " + rs1.getString("credits"));

        // WHERE course_id = ?
        System.out.println("\tdepartment list");
        stmt2.setString(1, course_id);
        try (ResultSet rs2 = stmt2.executeQuery()) {
            while (rs2.next()) {
                // SELECT dept_name
                System.out.println("\t\tname: " + rs2.getString("dept_name"));
            }
        }
    }
}
}

```

다음은 수업시간표 및 강의실 조회를 위한 두 번째 기능을 구현한 함수의 핵심 로직이다 사용자로부터 연도와 학기를 입력 받고, 해당 연도와 학기에 개설된 수업의 강좌 ID, 강좌 제목, 섹션 ID 를 데이터베이스에서 조회하고 출력한다. 이를 위해 stmt1 을 사용하여 첫 번째 쿼리를 준비하고 반복 실행한다. 조회된 각 수업에 대하여 stmt2 를 설정하고 두 번째 쿼리를 반복 실행한다. 특정 강좌 ID 와 섹션 ID 에 해당하는 수업의 요일, 시작 시간, 종료 시간, 건물명, 강의실 번호를 리스트로 출력할 수 있다.

```

PreparedStatement stmt1 = conn.prepareStatement(query1);
PreparedStatement stmt2 = conn.prepareStatement(query2);

System.out.print("연도 학기: ");
int year = sc.nextInt();
String semester = sc.next();
// WHERE (S.year, S.semester) = (?, ?)
stmt1.setInt(1, year);
stmt1.setString(2, semester);

try (ResultSet rs1 = stmt1.executeQuery()) {
    while (rs1.next()) {
        // SELECT S.course_id, C.title, S.sec_id
        String course_id = rs1.getString("S.course_id");
        String sec_id = rs1.getString("S.sec_id");
        System.out.print("course id: " + course_id);
        System.out.print(" | title: " + rs1.getString("C.title"));
        System.out.println(" | section id: " + sec_id);
    }
}
}

```

```

    // AND (R.course_id, R.sec_id, R.year, R.semester) = (?, ?, ?, ?)
    pstmt2.setString(1, course_id);
    pstmt2.setString(2, sec_id);
    pstmt2.setInt(3, year);
    pstmt2.setString(4, semester);
    try (ResultSet rs2 = pstmt2.executeQuery()) {
        while (rs2.next()) {
            // SELECT T.day, T.start_time, T.end_time, R.building, R.room_number
            System.out.print("\tL day: " + rs2.getString("T.day") + ", ");
            System.out.print("start time: " + rs2.getString("T.start_time") + ", ");
            System.out.print("end time: " + rs2.getString("T.end_time") + ", ");
            System.out.print("building: " + rs2.getString("R.building") + ", ");
            System.out.println("room no: " + rs2.getString("R.room_number"));
        }
    }
}
}

```

다음은 프로젝트 수행결과 조회를 위한 세 번째 기능을 구현한 함수의 핵심 로직이다. 우선 학번, 연도, 학기를 입력 받는다. 이 정보는 프로젝트 조회에 필요한 파라미터로 활용된다. pstmt1, pstmt2, pstmt3, pstmt4 를 각각의 SQL 쿼리와 함께 초기화한다. 이 쿼리들은 위의 SQL 설명과 같이 학생 정보, 수강 정보, 프로젝트 세부 사항 및 프로젝트 성적의 평균과 총계를 출력하는데 사용된다. 입력된 값을 기반으로 pstmt1 과 pstmt2 를 실행하여 학생의 ID, 이름, 소속 학과 이름과 해당 학생이 특정 연도의 학기에 수강한 강좌의 ID, 제목, 섹션 ID, 그리고 성적을 반복적으로 출력한다. 해당 결과로 나온 각 section 의 정보를 이용하여, pstmt3 를 실행하여 학생이 참여한 프로젝트 번호, 프로젝트 이름, 최대 점수, 지도 교수 이름, 학생의 프로젝트 점수를 반복 출력한다. 마지막으로 pstmt4 를 실행하여 학생이 참여한 프로젝트의 총 개수와 평균 점수를 출력한다.

```

PreparedStatement pstmt1 = conn.prepareStatement(query1);
PreparedStatement pstmt2 = conn.prepareStatement(query2);
PreparedStatement pstmt3 = conn.prepareStatement(query3);
PreparedStatement pstmt4 = conn.prepareStatement(query4);

System.out.print("L 연도 학기 학번: ");
int year = sc.nextInt();
String semester = sc.next();
String s_id = sc.next();

// WHERE ID = ?
	pstmt1.setString(1, s_id);
// AND (T.ID, T.year, T.semester) = (?, ?, ?)
	pstmt2.setString(1, s_id);
	pstmt2.setInt(2, year);
	pstmt2.setString(3, semester);

try (ResultSet rs1 = pstmt1.executeQuery()) {
    // SELECT ID, name, dept_name
    rs1.next();
    System.out.print("ID: " + rs1.getString("ID"));
    System.out.print(", name: " + rs1.getString("name"));
    System.out.println(", dept name: " + rs1.getString("dept_name"));
}
}
}

```

```

try (ResultSet rs2 = pstmt2.executeQuery()) {
    while (rs2.next()) {
        // SELECT T.course_id, C.title, T.sec_id, T.grade
        String course_id = rs2.getString("T.course_id");
        String sec_id = rs2.getString("T.sec_id");
        System.out.print("\t\t course id: " + course_id);
        System.out.print(", title: " + rs2.getString("C.title"));
        System.out.print(", sec id: " + sec_id);
        System.out.println(", grade: " + rs2.getString("T.grade"));

        try {
            // AND (P.s_id, P.course_id, P.sec_id, P.year, P.semester) = (?, ?, ?, ?, ?)
            pstmt3.setString(1, s_id);
            pstmt3.setString(2, course_id);
            pstmt3.setString(3, sec_id);
            pstmt3.setInt(4, year);
            pstmt3.setString(5, semester);
            try (ResultSet rs3 = pstmt3.executeQuery()) {
                while (rs3.next()) {
                    // P.num, P.name, P.max_score, I.name, P.score
                    System.out.print("\t\t\t project num: " +
rs3.getString("P.num"));
                    System.out.print(", name: " + rs3.getString("P.name"));
                    System.out.print(", score: " + rs3.getInt("P.score"));
                    System.out.print("/" + rs3.getInt("P.max_score"));
                    System.out.println(", instructor name: " +
rs3.getString("I.name"));
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        try {
            // WHERE (s_id, course_id, sec_id, year, semester) = (?, ?, ?, ?, ?, ?)
            pstmt4.setString(1, s_id);
            pstmt4.setString(2, course_id);
            pstmt4.setString(3, sec_id);
            pstmt4.setInt(4, year);
            pstmt4.setString(5, semester);
            try (ResultSet rs4 = pstmt4.executeQuery()) {
                // SELECT count(*) as count, avg(score) as avg
                rs4.next();
                System.out.print("\t\t** total count: " + rs4.getInt("count"));
                System.out.println(", average score: " + rs4.getInt("avg") + " **");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}

```

5. DB 데이터

다음과 같다.

```
1 •  SELECT *
2   FROM course
```

Result Grid Filter Rows:

course_id	title	credits
CSE101	Computer Science I	4
ENG101	English Literature	3
HIS101	World History	3
MAT101	Calculus I	3
NULL	NULL	NULL

```
1 •  SELECT *
2   FROM section
```

Result Grid Filter Rows:

course_id	sec_id	semester	year
CSE101	S0	Fall	2022
CSE101	S0	Winter	2023
CSE101	S1	Winter	2023
ENG101	S1	Spring	2021
HIS101	S1	Fall	2022
MAT101	S0	Fall	2022
NULL	NULL	NULL	NULL

```
1 •  SELECT *
2   FROM course_dept
```

Result Grid Filter Rows:

course_id	dept_name
CSE101	Computer Science Dept
CSE101	English Dep
CSE101	Mathematics Dept
ENG101	English Dept
ENG101	History Dept
HIS101	History Dept
MAT101	Computer Science Dept
MAT101	Mathematics Dept
NULL	NULL

```
1 •  SELECT *
2   FROM time_slot
```

Result Grid Filter Rows:

time_slot_...	day	start_time	end_time
A	Fri	15	16
A	Mon	11	13
A	Wen	13	14
B	Tue	9	11
B	Tue	15	17
C	Sat	10	13
NULL	NULL	NULL	NULL

```
1 •  SELECT *
2   FROM student
```

Result Grid Filter Rows:

ID	name	dept_name	tot_cred
01111	Minseok	Computer Science Dept	73
02345	Smith	History Dept	28
04567	Bob	Mathematics Dept	16
06789	Kiyoun	Computer Science Dept	36
07890	Alex	English Dep	27
NULL	NULL	NULL	NULL

```
1 •  SELECT *
2   FROM instructor
```

Result Grid Filter Rows:

ID	name	dept_name	salary
11234	Dr. Peak	Mathematics Dept	75000.00
12345	Dr. Son	Computer Science Dept	72000.00
15678	Dr. Lee	English Dep	80000.00
16789	Dr. Kang	Computer Science Dept	82000.00
17890	Dr. Kim	History Dept	78000.00
NULL	NULL	NULL	NULL

```
1 •  SELECT *
2   FROM takes
```

Result Grid Filter Rows:

ID	course_id	sec_id	semester	year	grade
01111	CSE101	S0	Fall	2022	C-
01111	CSE101	S0	Winter	2023	A
01111	HIS101	S1	Fall	2022	B+
01111	MAT101	S0	Fall	2022	A
02345	HIS101	S1	Fall	2022	B+
04567	MAT101	S0	Fall	2022	A-
06789	CSE101	S1	Winter	2023	B
07890	ENG101	S1	Spring	2021	C
NULL	NULL	NULL	NULL	NULL	NULL

```

1 •   SELECT *
2     FROM rooms

```

0% 11:2

Result Grid Filter Rows: Search Export:

course_id	sec_id	year	semester	time_slot...	day	start_time	building	room_number
CSE101	S0	2023	Winter	A	Mon	11	310	723
CSE101	S0	2023	Winter	A	Wen	13	310	727
CSE101	S0	2023	Winter	A	Fri	15	208	101
CSE101	S1	2023	Winter	B	Tue	9	207	212
CSE101	S1	2023	Winter	B	Tue	15	310	727
MAT101	S0	2022	Fall	B	Tue	9	301	123
MAT101	S0	2022	Fall	B	Tue	15	107	111
HIS101	S1	2022	Fall	B	Tue	9	208	109
HIS101	S1	2022	Fall	B	Tue	15	208	108
CSE101	S0	2022	Fall	A	Mon	11	310	721
CSE101	S0	2022	Fall	A	Wen	13	310	B601
CSE101	S0	2022	Fall	A	Fri	15	207	111
ENG101	S1	2021	Spring	C	Sat	10	310	519

```

1 •   SELECT *
2     FROM project

```

100% 13:2

Result Grid Filter Rows: Search Edit: Export/Import:

s_id	course_id	sec_id	semester	year	num	name		i_id	max_score	score
01111	CSE101	S0	Fall	2022	1	Data Structures Project I		16789	100	11
01111	CSE101	S0	Fall	2022	2	Algorithm Design I		15678	100	20
01111	CSE101	S0	Winter	2023	1	Data Structures Project I		16789	50	50
01111	CSE101	S0	Winter	2023	2	Algorithm Design I		15678	50	45
01111	HIS101	S1	Fall	2022	1	Historical Research		17890	100	35
01111	MAT101	S0	Fall	2022	1	Basic Algebra I		16789	100	35
01111	MAT101	S0	Fall	2022	2	Linear Algebra Study		11234	100	93
02345	HIS101	S1	Fall	2022	1	Historical Research		17890	100	82
04567	MAT101	S0	Fall	2022	1	Advanced Calculus Project		11234	100	88
06789	CSE101	S1	Winter	2023	1	Algorithm Design I		16789	100	85
07890	ENG101	S1	Spring	2021	1	Literary Analysis		15678	100	78
NULL	NULL	NULL	NULL	NULL	NULL	NULL		NULL	NULL	NULL

6. Java 프로그램 실행 결과

1번 기능

```
↑ 조회(1:융합과목, 2.수업 시간 및 강의실, 3.프로젝트 수행결과, 4.종료): 1
↓ 연도 학기: 2022 fall
└ course id: CSE101 | title: Computer Science I | credits: 4
    department list
        L name: Computer Science Dept
        L name: English Dep
        L name: Mathematics Dept
    course id: MAT101 | title: Calculus I | credits: 3
    department list
        L name: Computer Science Dept
        L name: Mathematics Dept
```

2번 기능

```
↑ 조회(1:융합과목, 2.수업 시간 및 강의실, 3.프로젝트 수행결과, 4.종료): 2
↓ 연도 학기: 2022 fall
└ course id: CSE101 | title: Computer Science I | section id: S0
    L day: Mon, start time: 11, end time: 13, building: 310, room no: 721
    L day: Wen, start time: 13, end time: 14, building: 310, room no: B601
    L day: Fri, start time: 15, end time: 16, building: 207, room no: 111
└ course id: HIS101 | title: World History | section id: S1
    L day: Tue, start time: 9, end time: 11, building: 208, room no: 109
    L day: Tue, start time: 15, end time: 17, building: 208, room no: 108
└ course id: MAT101 | title: Calculus I | section id: S0
    L day: Tue, start time: 9, end time: 11, building: 301, room no: 123
    L day: Tue, start time: 15, end time: 17, building: 107, room no: 111
```

3번 기능

```
↑ 조회(1:융합과목, 2.수업 시간 및 강의실, 3.프로젝트 수행결과, 4.종료): 3
↓ 연도 학기 학번: 2022 fall 01111
ID: 01111, name: Minseok, dept name: Computer Science Dept
    course id: CSE101, title: Computer Science I, sec id: S0, grade: C-
        L project num: 1, name: Data Structures Project I, score: 11/100, instructor name: Dr. Kang
        L project num: 2, name: Algorithm Design I, score: 20/100, instructor name: Dr. Lee
        ** total count: 2, average score: 15 **
    course id: HIS101, title: World History, sec id: S1, grade: B+
        L project num: 1, name: Historical Research, score: 35/100, instructor name: Dr. Kim
        ** total count: 1, average score: 35 **
    course id: MAT101, title: Calculus I, sec id: S0, grade: A
        L project num: 1, name: Basic Algebra I, score: 35/100, instructor name: Dr. Kang
        L project num: 2, name: Linear Algebra Study, score: 93/100, instructor name: Dr. Peak
        ** total count: 2, average score: 64 **
```

7. 정확성 검증

1번 기능 정확성 검증:

```
↑ 조회(1:융합과목, 2.수업 시간 및 강의실, 3.프로젝트 수행결과, 4.종료): 1
↓ 연도 학기: 2022 fall
└ course id: CSE101 | title: Computer Science I | credits: 4
    department list
        L name: Computer Science Dept
        L name: English Dep
        L name: Mathematics Dept
└ course id: MAT101 | title: Calculus I | credits: 3
    department list
        L name: Computer Science Dept
        L name: Mathematics Dept
```

The screenshot shows two SQL queries in the SQL tab and their corresponding Result Grids.

Query 1 (Left):

```
1 • SELECT *
2   FROM section
```

Result Grid (Left):

course_id	sec_id	semester	year
CSE101	S0	Fall	2022
CSE101	S0	Winter	2023
CSE101	S1	Winter	2023
ENG101	S1	Spring	2021
HIS101	S1	Fall	2022
MAT101	S0	Fall	2022
NULL	NULL	NULL	NULL

Query 2 (Right):

```
1 • SELECT *
2   FROM course_dept
```

Result Grid (Right):

course_id	dept_name
CSE101	Computer Science Dept
CSE101	English Dep
CSE101	Mathematics Dept
ENG101	English Dept
ENG101	History Dept
HIS101	History Dept
MAT101	Computer Science Dept
MAT101	Mathematics Dept
NULL	NULL

section 테이블에서 2022년 fall 학기에 개설된 과목은 “CSE101”, “HIS101”, “MAT101”으로 총 3개이다. 이때 course_dept 테이블을 보면, “CSE101”, “MAT101”은 2개 이상의 학과에서 개설하는 융합과목이고, “HIS101”은 1개의 학과에서 개설하여 융합과목이 아니다. 따라서 결과 값으로, “CSE101”, “MAT101” 2개의 과목만 보여주고, 해당 과목들의 개설 학과를 리스트로 보여주는 위의 결과가 옳다.

2번 기능 정확성 검증

```
↑ 조회(1:융합과목, 2.수업 시간 및 강의실, 3.프로젝트 수행결과, 4.종료): 2
↓ 연도 학기: 2022 fall
└ course id: CSE101 | title: Computer Science I | section id: S0
    L day: Mon, start time: 11, end time: 13, building: 310, room no: 721
    L day: Wen, start time: 13, end time: 14, building: 310, room no: B601
    L day: Fri, start time: 15, end time: 16, building: 207, room no: 111
└ course id: HIS101 | title: World History | section id: S1
    L day: Tue, start time: 9, end time: 11, building: 208, room no: 109
    L day: Tue, start time: 15, end time: 17, building: 208, room no: 108
└ course id: MAT101 | title: Calculus I | section id: S0
    L day: Tue, start time: 9, end time: 11, building: 301, room no: 123
    L day: Tue, start time: 15, end time: 17, building: 107, room no: 111
```

The figure consists of three vertically stacked MySQL Workbench screenshots. The top two show the results of running queries:

```
1 • SELECT *
2   FROM section
```

course_id	sec_id	semester	year
CSE101	S0	Fall	2022
CSE101	S0	Winter	2023
CSE101	S1	Winter	2023
ENG101	S1	Spring	2021
HIS101	S1	Fall	2022
MAT101	S0	Fall	2022
NULL	NULL	NULL	NULL


```
1 • SELECT *
2   FROM time_slot
```

time_slot_...	day	start_time	end_time
A	Fri	15	16
A	Mon	11	13
A	Wen	13	14
B	Tue	9	11
B	Tue	15	17
C	Sat	10	13
NULL	NULL	NULL	NULL


```
1 • SELECT *
2   FROM rooms
```

course_id	sec_id	year	semester	time_slot_...	day	start_time	building	room_number
CSE101	S0	2023	Winter	A	Mon	11	310	723
CSE101	S0	2023	Winter	A	Wen	13	310	727
CSE101	S0	2023	Winter	A	Fri	15	208	101
CSE101	S1	2023	Winter	B	Tue	9	207	212
CSE101	S1	2023	Winter	B	Tue	15	310	727
MAT101	S0	2022	Fall	B	Tue	9	301	123
MAT101	S0	2022	Fall	B	Tue	15	107	111
HIS101	S1	2022	Fall	B	Tue	9	208	109
HIS101	S1	2022	Fall	B	Tue	15	208	108
CSE101	S0	2022	Fall	A	Mon	11	310	721
CSE101	S0	2022	Fall	A	Wen	13	310	B601
CSE101	S0	2022	Fall	A	Fri	15	207	111
ENG101	S1	2021	Spring	C	Sat	10	310	519

마찬가지로 section 테이블에서 2022년 fall 학기 개설된 과목은 “CSE101”, “HIS101”, “MAT101”으로 총 3개이다. “CSE101”는 A 유형으로 총 3개의 시간대에 개설된다. 각각 “Mon, 11~13, 310-721”, “Wen, 13~14, 310-B601”, “Fri, 15~16, 207-111”이고, 이 결과는 실제 프로그램 실행 결과로도 정확하게 표현되었다. 마찬가지로 나머지 “HIS101”, “MAT101” 과목도 정확하게 출력되는 것을 확인할 수 있다.

3번 기능 정확성 검증

```
 조회(1:융합과목, 2.수업 시간 및 강의실, 3.프로젝트 수행결과, 4.종료): 3
↳ 연도 학기 학번: 2022 fall 01111
ID: 01111, name: Minseok, dept name: Computer Science Dept
  ↳ course id: CSE101, title: Computer Science I, sec id: S0, grade: C-
    ↳ project num: 1, name: Data Structures Project I, score: 11/100, instructor name: Dr. Kang
    ↳ project num: 2, name: Algorithm Design I, score: 20/100, instructor name: Dr. Lee
    ** total count: 2, average score: 15 **
  ↳ course id: HIS101, title: World History, sec id: S1, grade: B+
    ↳ project num: 1, name: Historical Research, score: 35/100, instructor name: Dr. Kim
    ** total count: 1, average score: 35 **
  ↳ course id: MAT101, title: Calculus I, sec id: S0, grade: A
    ↳ project num: 1, name: Basic Algebra I, score: 35/100, instructor name: Dr. Kang
    ↳ project num: 2, name: Linear Algebra Study, score: 93/100, instructor name: Dr. Peak
    ** total count: 2, average score: 64 **
```

Three separate MySQL query results displayed in a grid interface:

- Result 1:** SELECT * FROM takes

ID	course_id	sec_id	semester	year	grade
01111	CSE101	S0	Fall	2022	C-
01111	CSE101	S0	Winter	2023	A
01111	HIS101	S1	Fall	2022	B+
01111	MAT101	S0	Fall	2022	A
02345	HIS101	S1	Fall	2022	B+
04567	MAT101	S0	Fall	2022	B+
06789	CSE101	S1	Winter	2023	B-
07890	ENG101	S1	Spring	2021	C
NULL	NULL	NULL	NULL	NULL	NULL

- Result 2:** SELECT * FROM section

course_id	sec_id	semester	year
CSE101	S0	Fall	2022
CSE101	S0	Winter	2023
CSE101	S1	Winter	2023
ENG101	S1	Spring	2021
HIS101	S1	Fall	2022
MAT101	S0	Fall	2022
NULL	NULL	NULL	NULL

- Result 3:** SELECT * FROM instructor

ID	name	dept_name	salary
11234	Dr. Peak	Mathematics Dept	75000.00
12345	Dr. Son	Computer Science Dept	72000.00
15678	Dr. Lee	English Dep	80000.00
16789	Dr. Kang	Computer Science Dept	82000.00
17890	Dr. Kim	History Dept	78000.00
NULL	NULL	NULL	NULL

MySQL query result for projects:

```
1 •  SELECT *
2   FROM project
```

100% 13:2

Result Grid Filter Rows: Search Edit: Export/Import:

s_id	course_id	sec_id	semester	year	num	name	i_id	max_score	score
01111	CSE101	S0	Fall	2022	1	Data Structures Project I	16789	100	11
01111	CSE101	S0	Fall	2022	2	Algorithm Design I	15678	100	20
01111	CSE101	S0	Winter	2023	1	Data Structures Project I	16789	50	50
01111	CSE101	S0	Winter	2023	2	Algorithm Design I	15678	50	45
01111	HIS101	S1	Fall	2022	1	Historical Research	17890	100	35
01111	MAT101	S0	Fall	2022	1	Basic Algebra I	16789	100	35
01111	MAT101	S0	Fall	2022	2	Linear Algebra Study	11234	100	93
02345	HIS101	S1	Fall	2022	1	Historical Research	17890	100	82
04567	MAT101	S0	Fall	2022	1	Advanced Calculus Project	11234	100	88
06789	CSE101	S1	Winter	2023	1	Algorithm Design I	16789	100	85
07890	ENG101	S1	Spring	2021	1	Literary Analysis	15678	100	78
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

01111(input) 학생이 2022 년 fall 학기에서 수강한 과목은 “CSE101”, “HIS101”, “MAT101” 총 3 개이다. 따라서 출력 값이 크게 3 가지 부분으로 나뉘는 것을 확인할 수 있다. 이제 project 테이블을 참고하여, “CSE101” 과목에서 수행한 프로젝트 목록을 보면, 총 2 개 임을 확인할 수 있다. 마찬가지로 해당 데이터가 정상적으로 프로그램 출력 결과로 표시되는 것을 확인할 수 있다. 또한 지도 교수의 ID 를 이용하여, 정상적으로 지도교수 이름을 출력하는 것도 확인할 수 있다. 마지막으로, 총 2 번의 프로젝트를 수행했다는 것과, 프로젝트의 평균 또한 집계함수를 통해 정상적으로 계산했음을 확인할 수 있다.

8. Java 프로그램 소스코드

```
package src;

import java.sql.*;
import java.util.Scanner;

public class Main {
    static final String DB_URL =
"jdbc:mysql://localhost/assignment13?useUnicode=true&useJDBCCompliantTimezoneShift=true&
useLegacyDatetimeCode=false&serverTimezone=UTC";
    static final String USER = "root";
    static final String PASS = "1234";
    static Connection conn;
    static final Scanner sc = new Scanner(System.in);

    public static void main(String[] args) {
        try {
            conn = DriverManager.getConnection(DB_URL, USER, PASS);
            int command;

            while (true) {
                System.out.print("조회(1:융합과목, 2.수업 시간 및 강의실, 3.프로젝트 수행결과,
4.종료): ");
                command = sc.nextInt();
                if (command == 1)
                    searchMixCourse();
                else if (command == 2)
                    sectionTimeAndClassroom();
                else if (command == 3)
                    searchProject();
                else if (command == 4)
                    break;
                System.out.println("\n");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            if (conn != null) {
                try {
                    conn.close();
                } catch (SQLException ex) {
                    ex.printStackTrace();
                }
            }
        }
    }

    static void searchMixCourse() {
        String query1 = """
            SELECT course_id, title, credits
            FROM course
            WHERE course_id IN (
                SELECT S.course_id
                FROM section as S, course_dept as CD
                WHERE S.course_dept_id = CD.course_dept_id
                GROUP BY S.course_id
                HAVING COUNT(S.section_id) > 1
            )
        """;
        Statement statement = null;
        ResultSet resultSet = null;
        try {
            statement = conn.createStatement();
            resultSet = statement.executeQuery(query1);
            if (resultSet != null) {
                while (resultSet.next()) {
                    System.out.println("과목ID: " + resultSet.getString("course_id") +
                        ", 과목명: " + resultSet.getString("title") +
                        ", 학점: " + resultSet.getString("credits"));
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            if (statement != null)
                statement.close();
            if (resultSet != null)
                resultSet.close();
        }
    }
}
```

```

        WHERE (S.year, S.semester) = (?, ?)
        AND S.course_id = CD.course_id
        GROUP BY S.course_id
        HAVING COUNT(*) >= 2
    )""";
String query2 = """
    SELECT dept_name
    FROM course_dept
    WHERE course_id = ?
""";
try {
    PreparedStatement pstmt1 = conn.prepareStatement(query1);
    PreparedStatement pstmt2 = conn.prepareStatement(query2);

    System.out.print("▶ 연도 학기: ");
    int year = sc.nextInt();
    String semester = sc.next();

    // WHERE (S.year, S.semester) = (?, ?)
    pstmt1.setInt(1, year);
    pstmt1.setString(2, semester);

    try (ResultSet rs1 = pstmt1.executeQuery()) {
        while (rs1.next()) {
            // SELECT course_id, title, credits
            String course_id = rs1.getString("course_id");
            System.out.print("\t course id: " + course_id);
            System.out.print("\t | title: " + rs1.getString("title"));
            System.out.println("\t | credits: " + rs1.getString("credits"));

            // WHERE course_id = ?
            System.out.println("\tdepartment list");
            pstmt2.setString(1, course_id);
            try (ResultSet rs2 = pstmt2.executeQuery()) {
                while (rs2.next()) {
                    // SELECT dept_name
                    System.out.println("\t\t name: " +
rs2.getString("dept_name"));
                }
            }
        }
    }
} catch (SQLException e) {
    e.printStackTrace();
}
}

static void sectionTimeAndClassroom() {
    String query1 = """
        SELECT S.course_id, C.title, S.sec_id
        FROM section as S, course as C
        WHERE (S.year, S.semester) = (?, ?)
        AND S.course_id = C.course_id
""";
}

```

```

String query2 = """
    SELECT T.day, T.start_time, T.end_time, R.building, R.room_number
    FROM rooms as R, time_slot as T
    WHERE (R.time_slot_id, R.day, R.start_time) = (T.time_slot_id, T.day,
T.start_time)
        AND (R.course_id, R.sec_id, R.year, R.semester) = (?, ?, ?, ?)
    """;
try {
    PreparedStatement pstmt1 = conn.prepareStatement(query1);
    PreparedStatement pstmt2 = conn.prepareStatement(query2);

    System.out.print("▶ 연도 학기: ");
    int year = sc.nextInt();
    String semester = sc.next();
    // WHERE (S.year, S.semester) = (?, ?)
    pstmt1.setInt(1, year);
    pstmt1.setString(2, semester);

    try (ResultSet rs1 = pstmt1.executeQuery()) {
        while (rs1.next()) {
            // SELECT S.course_id, C.title, S.sec_id
            String course_id = rs1.getString("S.course_id");
            String sec_id = rs1.getString("S.sec_id");
            System.out.print("▶ course id: " + course_id);
            System.out.print(" | title: " + rs1.getString("C.title"));
            System.out.println(" | section id: " + sec_id);

            // AND (R.course_id, R.sec_id, R.year, R.semester) = (?, ?, ?, ?)
            pstmt2.setString(1, course_id);
            pstmt2.setString(2, sec_id);
            pstmt2.setInt(3, year);
            pstmt2.setString(4, semester);
            try (ResultSet rs2 = pstmt2.executeQuery()) {
                while (rs2.next()) {
                    // SELECT T.day, T.start_time, T.end_time, R.building,
R.room_number
                    System.out.print("\t▶ day: " + rs2.getString("T.day") + ", ");
                    System.out.print("start time: " +
rs2.getString("T.start_time") + ", ");
                    System.out.print("end time: " + rs2.getString("T.end_time") +
", ");
                    System.out.print("building: " + rs2.getString("R.building") +
", ");
                    System.out.println("room no: " +
rs2.getString("R.room_number"));
                }
            }
        }
    }
} catch (SQLException e) {
    e.printStackTrace();
}
}

```

```

static void searchProject() {
    String query1 = """
        SELECT ID, name, dept_name
        FROM student
        WHERE ID = ?
        """;
    String query2 = """
        SELECT T.course_id, C.title, T.sec_id, T.grade
        FROM takes as T, course as C
        WHERE T.course_id = C.course_id
        AND (T.ID, T.year, T.semester) = (?, ?, ?)
        """;
    String query3 = """
        SELECT P.num, P.name, P.max_score, I.name, P.score
        FROM project as P, instructor as I
        WHERE P.i_id = I.id
        AND (P.s_id, P.course_id, P.sec_id, P.year, P.semester) = (?, ?, ?, ?, ?,
    )
    """;
    String query4 = """
        SELECT count(*) as count, avg(score) as avg
        FROM project
        WHERE (s_id, course_id, sec_id, year, semester) = (?, ?, ?, ?, ?, ?)
        """;
    try {
        PreparedStatement pstmt1 = conn.prepareStatement(query1);
        PreparedStatement pstmt2 = conn.prepareStatement(query2);
        PreparedStatement pstmt3 = conn.prepareStatement(query3);
        PreparedStatement pstmt4 = conn.prepareStatement(query4);

        System.out.print("γ 연도 학기 학번: ");
        int year = sc.nextInt();
        String semester = sc.next();
        String s_id = sc.next();

        // WHERE ID = ?
        pstmt1.setString(1, s_id);
        // AND (T.ID, T.year, T.semester) = (?, ?, ?)
        pstmt2.setString(1, s_id);
        pstmt2.setInt(2, year);
        pstmt2.setString(3, semester);

        try (ResultSet rs1 = pstmt1.executeQuery()) {
            // SELECT ID, name, dept_name
            rs1.next();
            System.out.print("ID: " + rs1.getString("ID"));
            System.out.print(", name: " + rs1.getString("name"));
            System.out.println(", dept name: " + rs1.getString("dept_name"));

            try (ResultSet rs2 = pstmt2.executeQuery()) {
                while (rs2.next()) {
                    // SELECT T.course_id, C.title, T.sec_id, T.grade
                    String course_id = rs2.getString("T.course_id");

```

```

        String sec_id = rs2.getString("T.sec_id");
        System.out.print("\t\t course id: " + course_id);
        System.out.print(", title: " + rs2.getString("C.title"));
        System.out.print(", sec id: " + sec_id);
        System.out.println(", grade: " + rs2.getString("T.grade"));

        try {
            // AND (P.s_id, P.course_id, P.sec_id, P.year, P.semester) =
            (?, ?, ?, ?, ?)
            pstmt3.setString(1, s_id);
            pstmt3.setString(2, course_id);
            pstmt3.setString(3, sec_id);
            pstmt3.setInt(4, year);
            pstmt3.setString(5, semester);
            try (ResultSet rs3 = pstmt3.executeQuery()) {
                while (rs3.next()) {
                    // P.num, P.name, P.max_score, I.name, P.score
                    System.out.print("\t\t\t project num: " +
rs3.getString("P.num"));
                    System.out.print(", name: " +
rs3.getString("P.name"));
                    System.out.print(", score: " +
rs3.getInt("P.score"));
                    System.out.print("/" + rs3.getInt("P.max_score"));
                    System.out.println(", instructor name: " +
rs3.getString("I.name"));
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        try {
            // WHERE (s_id, course_id, sec_id, year, semester) = (?, ?, ?, ?, ?)
            pstmt4.setString(1, s_id);
            pstmt4.setString(2, course_id);
            pstmt4.setString(3, sec_id);
            pstmt4.setInt(4, year);
            pstmt4.setString(5, semester);
            try (ResultSet rs4 = pstmt4.executeQuery()) {
                // SELECT count(*) as count, avg(score) as avg
                rs4.next();
                System.out.print("\t\t** total count: " +
rs4.getInt("count"));
                System.out.println(", average score: " +
rs4.getInt("avg") + " **");
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
} catch (SQLException e) {

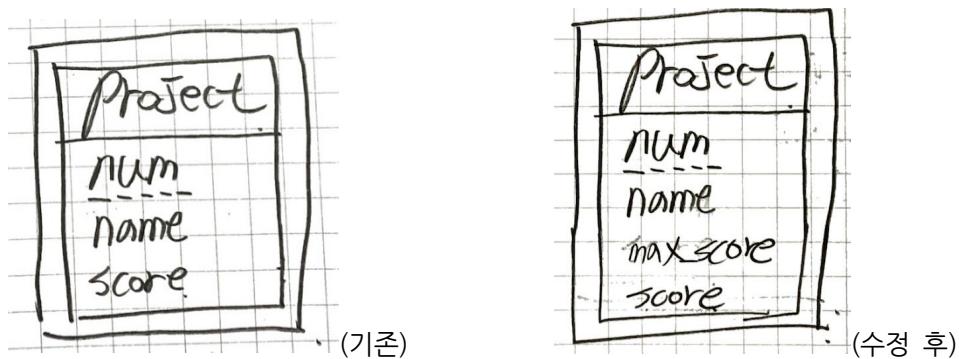
```

```
        e.printStackTrace();
    }
}
```

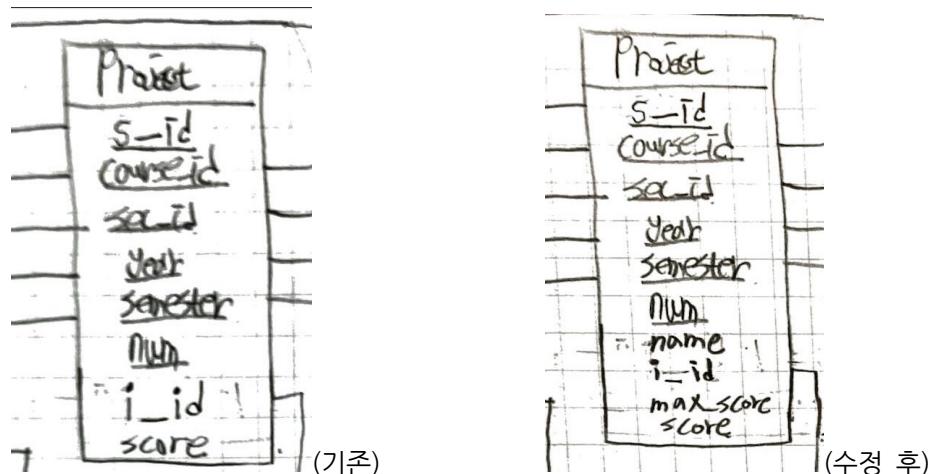
9. 기타

9.1. 누락을 확인한 부분: ERD의 project entity set을 테이블 스키마로 변환하는 과정에서, name과 max_score 속성을 옮기는 것이 누락되었다. 따라서, “3번 프로젝트 수행결과 조회” 과제를 수행함에 있어 해당 컬럼을 사용할 수 없게 되었고, 결과적으로 과제가 요구하는 프로젝트 이름과 배점을 출력하는 부분의 기능을 구현하는 것이 불가능하다.

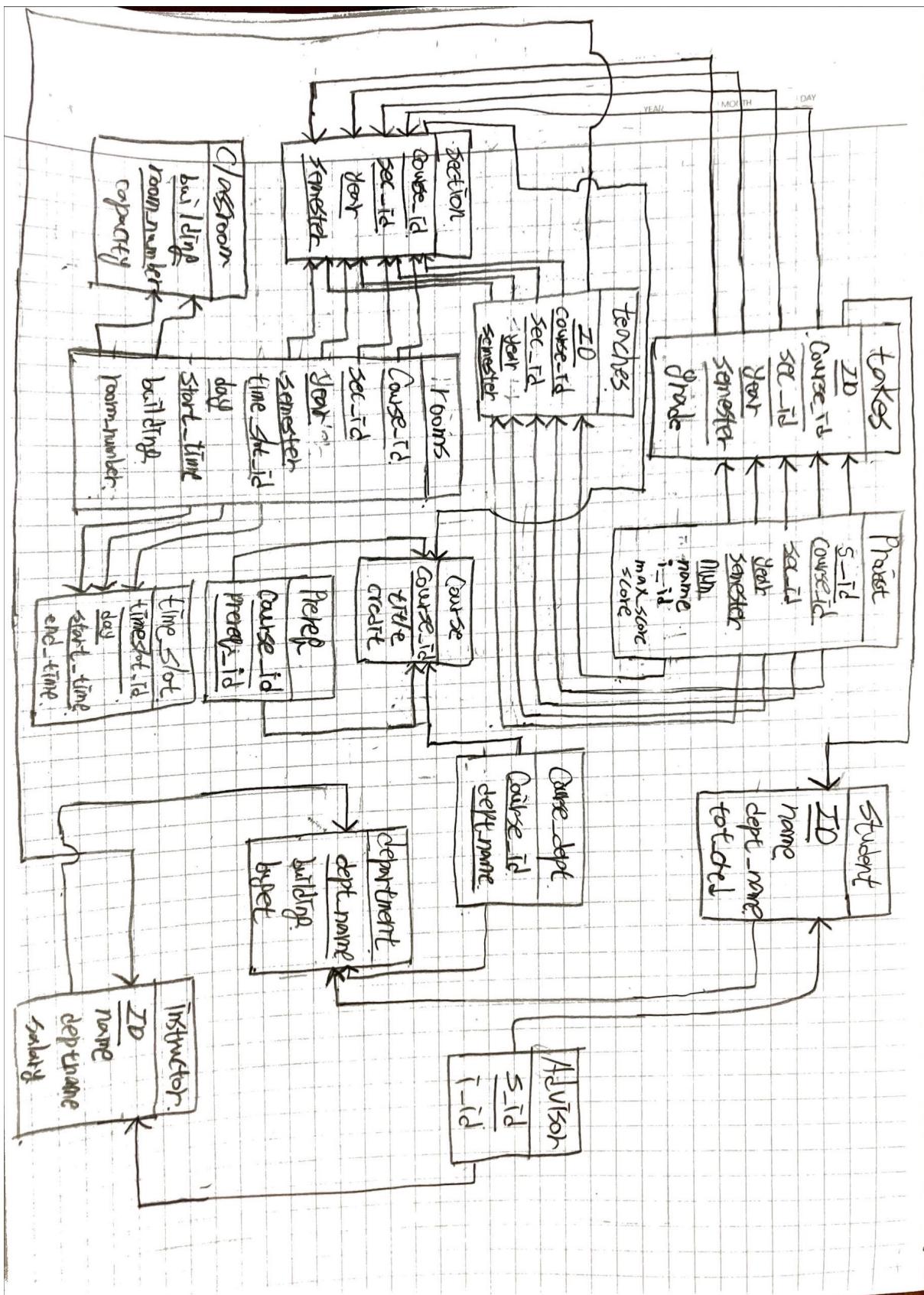
9.2. 누락 내용 기반 ER modeling의 수정: project entity set에 max_score(배점) 속성을 추가하였다. 기존 project entity set에서 해당 속성이 누락되어 최종 테이블 스키마에도 누락되었기에 다음과 같이 수정하였다.



9.3. 기존과 같은 변환규칙에 의한 테이블 스키마 변환의 수정: weak entity set이었던 project를 새롭게 추가된 max_score 속성과 기존 변환에서 누락된 name 속성까지 모두 포함해서 변환한다.



9.4. 수정된 RDB 스키마 다이어그램: 다음 장의 이미지와 같다.



최종 스키마: 최종적으로 완성된 스키마는 다음과 같다.

Department (dept_name, building, budget)

Course (course_id, title, credits)

Classroom (building, room_number, capacity)

Time_slot (time_slot_id, day, start_time, end_time)

Instructor (ID, name, salary, dept_name)

foreign key (dept_name) references Department (dept_name)

Student (ID, name, tot_cred, dept_name)

foreign key (dept_name) references Department (dept_name)

Teaches (ID, course_id, sec_id, year, semester)

foreign key (ID) references Instructor (ID)

foreign key (course_id, sec_id, year, semester) references Section (course_id, sec_id, year, semester)

Takes (ID, course_id, sec_id, year, semester, grade)

foreign key (ID) references Student (ID)

foreign key (course_id, sec_id, year, semester) references Section (course_id, sec_id, year, semester)

Project (s_id, course_id, sec_id, year, semester, num, name, i_id, max_score, score)

foreign key (s_id, course_id, sec_id, year, semester) references Takes (ID, course_id, sec_id, year, semester)

foreign key (i_id, course_id, sec_id, year, semester) references Teaches (ID, course_id, sec_id, year, semester)

Section (course_id, sec_id, year, semester)

foreign key (course_id) references Course (course_id)

Course_dept (course_id, dept_name)

foreign key (course_id) references Course (course_id)

foreign key (dept_name) references Department (dept_name)

Prereq (course_id, prereq_id)

foreign key (course_id) references Course (course_id)

foreign key (Prereq_id) references Course (course_id)

Advisor (s_id, i_id)

foreign key (s_id) references Student (ID)

foreign key (i_id) references Instructor (ID)

Rooms (course_id, sec_id, year, semester, time_slot_id, day, start_time, building, room_number)

foreign key (building, room_number) references Classroom (building, room_number)

10. 설계 수정

없음.