DB Object 스크립트(PL/SQL)

2조 박지명, 김민도, 조영선, 홍태훈, 황윤재

작성일자: 2026.02.10

목 차

A. 함수

1. 관리자 기능

요구사항번호		작성일	2026-02-09
요구사항명	주민번호 기반 성별 자동 추출	작성자	황윤재

```
-- 주민번호 뒷자리로 성별을 자동 추출하는 함수

-- 주민번호 기반 성별 추출 함수
CREATE OR REPLACE FUNCTION FN_GET_GENDER (
    p_ssn VARCHAR2 -- 주민번호 (예: 990101-1234567)
) RETURN VARCHAR2
IS
    v_gender_code CHAR(1);
    v_gender VARCHAR2(10);
BEGIN
    -- 주민번호 뒷자리 첫 번째 숫자 추출 (하이픈 포함 8번째)
    v_gender_code := SUBSTR(p_ssn, 8, 1);

    IF v_gender_code IN ('1', '3') THEN
        v_gender := '남자';
    ELSE
        v_gender := '여자';
    END IF;

    RETURN v_gender;
END;
/

-- [사용 예시]
SELECT
    u.usersName AS 이름,
    u.usersSSN AS 주민번호,
```

```
   -- 여기서 함수 호출! (DB에 저장된 게 아니라, 즉석에서 계산된 결과)
   FN_GET_GENDER(u.usersSSN) AS 성별
FROM STUDENT s
INNER JOIN USERS u ON s.usersSeq = u.usersSeq;
```

요구사항번호		작성일	2026-02-09
요구사항명	우수 교사 선정	작성자	황윤재

```
-- 교사 평가의 평균 점수가 9점 이상인
-- 교사를 우수 교사로 선정하는 함수

-- 우수 교사 선정 함수
CREATE OR REPLACE FUNCTION FN_CHECK_EXCELLENT_TEACHER (
   p_instructorSeq IN NUMBER
) RETURN VARCHAR2
IS
   v_avg_score NUMBER := 0;      -- 평균 점수를 담을 변수
   v_result VARCHAR2(20);        -- 결과 문자열 ('우수 교사' 등)
BEGIN
   -- 1. 해당 교사의 평가 점수 평균 계산
   SELECT AVG(tr.teacherRatingScore)
   INTO v_avg_score
   FROM TEACHER_RATING tr
   INNER JOIN COURSE_INSTRUCTOR ci ON tr.courseInstructorSeq =
ci.courseInstructorSeq
   INNER JOIN AFFILIATED_INSTRUCTOR ai ON ci.affiliatedInstructorSeq =
ai.affiliatedInstructorSeq
   WHERE ai.instructorSeq = p_instructorSeq;

   -- 2. 점수가 없으면(NULL) 0점으로 처리
   v_avg_score := NVL(v_avg_score, 0);

   -- 3. 판별 과정 (9점 이상이면 우수)
```

```
    IF v_avg_score >= 9 THEN
        v_result := '우수 교사';
    ELSE
        v_result := '일반 교사';
    END IF;

    RETURN v_result;
END;
/

-- 함수 사용 예시
SELECT
    i.instructorName AS 교사명,
    i.instructorTel AS 연락처,
    -- 여기서 함수를 호출해서 상태를 출력
    FN_CHECK_EXCELLENT_TEACHER(i.instructorSeq) AS 교사등급
FROM INSTRUCTOR i
WHERE FN_CHECK_EXCELLENT_TEACHER(i.instructorSeq) = '우수 교사';

commit;

select * from test_score;
```

요구사항번호		작성일	2026-02-09
요구사항명	훈련지원금 신청가능자 취합	작성자	김민도

```
-- function 훈련지원금 신청가능자
-- 목적: 오늘 날짜와 비교하여 훈련지원금 신청해야 하는 사람들을 보여준다
-- 작성자: 김민도

-- 비고: 훈련지원금은 수강할 과정이 시작되면 바로 신청할 수 있고 일회성 지급이며,
과정이 끝나면 소급신청 불가능하다
```

```
--     이미 신청했지만 미지급인 사람들은 보여주지 않는다

-- 신청 가능하지만 신청하지 않는 사람만 보여준다(현재DB에 등록된 상태)
create or replace function func_apply_allowance
return SYS_REFCURSOR
is
   v_cursor SYS_REFCURSOR;
begin
   open v_cursor for
      select
         S.STUDENTSEQ as 학생번호,
         U.USERSNAME as 이름,
         S.STUDENTSTATUS as 수강상태,
         RC.REGISTEREDCOURSESTARTDATE as 과정시작일,
         RC.REGISTEREDCOURSEENDDATE as 과정종료일
      from STUDENT S
         join USERS U
            on S.USERSSEQ = U.USERSSEQ
         join REGISTERED_COURSE RC
            on RC.REGISTEREDCOURSESEQ = S.REGISTEREDCOURSESEQ
      where TRUNC(SYSDATE) between RC.REGISTEREDCOURSESTARTDATE and
RC.REGISTEREDCOURSEENDDATE
        and S.STUDENTSTATUS = '수강중'
        and NOT EXISTS(
           select 1
           from TRAINING_ALLOWANCE TA
           where TA.STUDENTSEQ = S.STUDENTSEQ
      );

   return v_cursor;
end;
/
-- 확인용
select func_apply_allowance() from dual;
```

요구사항번호		작성일	2026-02-09
요구사항명		작성자	황윤재

```
-- 강의를 수료한 수강생 중 취업에 성공했고,
-- 강의 수료일에서 6개월이 지났지만 취업 성공 수당이
-- 지급되지 않은 수강생의 목록을 보는 함수

-- 취업 성공 수당 미지급 학생 조회 함수
create or replace function FN_FIND_UNPAID_GRADUATES
(
    p_studentSeq IN NUMBER
) return varchar2

is
    v_count number;

    -- (이미 받은 사람, 취업 못한 사람, 아직 6개월 안 지난 사람)
    v_result varchar2(1) := 'N';
begin
    -- 조건 검사
    select count(*)
    into v_count
    from student s

    -- 수강생(student)과 등록된 과정(registered_course) join
    join REGISTERED_COURSE rc on s.REGISTEREDCOURSESEQ =
rc.REGISTEREDCOURSESEQ

    -- 수강생(student)과 취업상태(employment_status) join
    join EMPLOYMENT_STATUS es on s.STUDENTSEQ = es.STUDENTSEQ

    -- 수강생(student)과 취업 성공 수당(employment_status) left join
```

```
   -- 수당을 신청하지 않은 수강생을 찾을 수 있도록 left join
   left join EMPLOYMENT_SUCCESS_ALLOWANCE esa on s.STUDENTSEQ =
esa.STUDENTSEQ

   -- s.STUDENTSEQ: FROM STUDENT s에서 가져온 실제 데이터들
   -- p_studentSeq: 사용자가 함수를 실행할 때 FN_FIND... (70) 하고 던져준 숫자 70
   where s.STUDENTSEQ = p_studentSeq
      and s.STUDENTSTATUS = '수강완료' -- 수료생
      and es.EMPLOYMENTSTATUSSTATE = '취업성공' -- 수료생 중에서 취업성공

      -- 수료생 중에서 취업성공하고 6개월 경과
      and rc.REGISTEREDCOURSEENDDATE <= add_months(sysdate, -6)

      -- 수료생 중에서 취업성공하고 6개월 경과됐는데 수당 미지급
      and (esa.employmentSuccessAllowanceState IS NULL OR
esa.employmentSuccessAllowanceState != '지급완료');

   -- v_count가 0보다 크다면? (조건에 맞는 데이터가 1건이라도 발견되었다면?)
   IF v_count > 0 THEN
      v_result := 'Y'; -- 이 학생은 대상자다(Y)
   ELSE
      v_result := 'N'; -- 이 학생은 대상자가 아니다(N)
   END IF;

   -- 최종 결과('Y' 또는 'N')를 함수 밖으로 던져줌
   RETURN v_result;



end;
/

-- 함수 사용 예시
SELECT
   u.usersName AS 학생이름,
```

```
    u.usersTel AS 전화번호,
    c.courseName AS 과정명,
    es.employmentStatusDate AS 취업일
FROM STUDENT s
JOIN USERS u ON s.usersSeq = u.usersSeq
JOIN REGISTERED_COURSE rc ON s.registeredCourseSeq = rc.registeredCourseSeq
JOIN COURSE c ON rc.courseSeq = c.courseSeq
JOIN EMPLOYMENT_STATUS es ON s.studentSeq = es.studentSeq
WHERE FN_IS_ALLOWANCE_TARGET(s.studentSeq) = 'Y'; -- 함수 호출
```

2. 교사 기능

요구사항번호		작성일	2026-02-09
요구사항명	기간별 출결 조회	작성자	홍태훈

```
-- 년, 월, 일별 출결 현황 조회
CREATE OR REPLACE FUNCTION fnGetAttendanceList (
    p_date IN VARCHAR2
) RETURN SYS_REFCURSOR
IS
    v_cursor SYS_REFCURSOR; -- 빨대
BEGIN
    OPEN v_cursor FOR
        SELECT
            학생이름,
            과정명,
            TO_CHAR(출석일, 'YYYY-MM-DD') AS 날짜,
            근태종류,
            출석률
        FROM VW_STUDENT_ATTENDANCE_STATUS
        WHERE TO_CHAR(출석일, 'YYYY-MM-DD') LIKE p_date || '%'
        ORDER BY 출석일 DESC, 학생이름;


    -- 함수라 반환
```

```
    RETURN v_cursor;
END;
/

--실행 코드
SELECT fnGetAttendanceList('2026-02') FROM DUAL;
```

요구사항번호		작성일	2026-02-09
요구사항명	결석일 계산	작성자	김민도

```
-- function 결석일 계산기
 -- 목적: 지각, 외출, 조퇴 3회시 결석 1회로 산정하여 총 결석일 수를 조회한다.
 -- 작성자: 김민도

CREATE OR REPLACE FUNCTION func_calculate_absence (
    p_student_seq IN NUMBER
) RETURN NUMBER
IS
    v_pure_absence  NUMBER := 0;
    v_other_bad_cnt NUMBER := 0;
    v_final_absence NUMBER := 0;
BEGIN
    -- 근태 데이터 집계
    SELECT
        COUNT(DECODE(AT.attendanceTypeState, '결석', 1)),
        COUNT(CASE WHEN AT.attendanceTypeState IN ('지각', '조퇴', '외출') THEN 1
END)
    INTO v_pure_absence, v_other_bad_cnt
    FROM ATTENDANCE A
    JOIN ATTENDANCE_TYPE AT ON A.attendanceTypeSeq = AT.attendanceTypeSeq
    WHERE A.studentSeq = p_student_seq;
```

```
    -- 3회당 1회 로직 적용
    v_final_absence := v_pure_absence + TRUNC(v_other_bad_cnt / 3);

    RETURN v_final_absence;
END;
/

-- 확인용
SELECT
    '학생번호: ' || 50 AS 정보,
    func_calculate_absence(50) AS "최종 계산된 결석일"
FROM DUAL;
```

B. 뷰

1. 관리자 기능

요구사항번호		작성일	2026-02-09
요구사항명	권한별 정보 조회	작성자	박지명

```
-- 뷰
-- 모든 학생(유저)의 정보를 출력하는 뷰 관리자가 목록보는 뷰임

CREATE OR REPLACE VIEW vw_user_course_info AS
SELECT DISTINCT
    u.usersseq,
    u.usersname as 학생이름,
    u.usersID as 학생아이디,
    u.userstel,
    st.studentseq,
    st.studentstatus,
    rc.registeredcourseseq,
    c.courseseq,
```

```
    c.coursename,
    i.INSTRUCTORNAME as 담당교사
FROM users u
    LEFT JOIN student st ON u.USERSSEQ = st.USERSSEQ
    LEFT JOIN REGISTERED_COURSE rc ON st.REGISTEREDCOURSESEQ =
rc.REGISTEREDCOURSESEQ
    LEFT JOIN course c ON rc.COURSESEQ = c.COURSESEQ
    INNER JOIN COURSE_INSTRUCTOR ci ON rc.REGISTEREDCOURSESEQ =
ci.REGISTEREDCOURSESEQ
    INNER JOIN AFFILIATED_INSTRUCTOR ai ON ai.AFFILIATEDINSTRUCTORSEQ =
ci.AFFILIATEDINSTRUCTORSEQ
    INNER JOIN INSTRUCTOR i ON i.INSTRUCTORSEQ = ai.INSTRUCTORSEQ;

-- 관리자가 모든 정보를 봄
select * from vw_user_course_info;
-- 특정교사가 자신들의 학생들의 정보를 봄
select * from vw_user_course_info
        where 담당교사 = '허이면';
-- 특정 교육생이 자신의 정보만 봄;
select * from vw_user_course_info
       where 학생아이디 = 'ebe0m9';
```

요구사항번호		작성일	2026-02-09
요구사항명	컴퓨터 관리	작성자	김민도

```
-- view 창고컴퓨터
-- 목적: 창고에 컴퓨터가 몇대 보관중인지 보여준다
-- 작성자: 김민도

create or replace view vw_storage_computer
as
select
```

```
    C.COMPUTERSEQ as 컴퓨터번호,
    CS.COMPUTERSTATUSSTATE as 컴퓨터상태,
    C.USEDCOMPUTER as 사용여부
from COMPUTER C
    join COMPUTER_STATUS CS
        on C.computerStatusSeq = CS.computerStatusSeq
where C.USEDCOMPUTER = '미사용';

select * from vw_storage_computer;
```

요구사항번호		작성일	2026-02-09
요구사항명		작성자	홍태훈

```
-- view 취업 성공 실시간 뷰

CREATE OR REPLACE VIEW vwSuccessJob AS
SELECT
    us.usersName AS 학생이름,
    es.employmentStatusDate AS 취업확정일,

    CASE
        WHEN es.employmentStatusDate IS NOT NULL AND es.employmentStatusDate <=
SYSDATE THEN '취업성공'
    END AS 취업상태_실시간

FROM STUDENT st
JOIN USERS us ON us.usersSeq = st.usersSeq
JOIN EMPLOYMENT_STATUS es ON es.STUDENTSEQ = st.STUDENTSEQ
WHERE es.employmentStatusDate IS NOT NULL
AND es.employmentStatusDate <= SYSDATE;

drop view vwSuccessJob;
```

```
-- 뷰 실행
SELECT * FROM vwSuccessJob;

-- 업데이트로 데이터 바꿔보기
UPDATE EMPLOYMENT_STATUS
SET employmentStatusDate = '2026-03-03'
WHERE studentSeq = (
    SELECT studentSeq
    FROM STUDENT
    WHERE usersSeq = 1
);
```

2. 교사 기능

요구사항번호		작성일	2026-02-09
요구사항명	학생 성적 조회	작성자	박지명

```
-- 학생 성적
CREATE OR REPLACE VIEW vw_student_score AS
SELECT
    st.studentseq,
    st.usersseq,
    u.USERSNAME,
    t.testseq,
    sub.SUBJECTTITLE,
    os.openedsubjectseq,
    t.TESTTYPE,
    ts.TESTGRADESCORE,
    ts.TESTGRADEPARTICIPATED
FROM student st
    INNER JOIN REGISTERED_COURSE rc ON st.REGISTEREDCOURSESEQ =
rc.REGISTEREDCOURSESEQ
    INNER JOIN OPENED_SUBJECT os ON rc.REGISTEREDCOURSESEQ =
```

```
os.REGISTEREDCOURSESEQ
    INNER JOIN subject sub ON os.SUBJECTSEQ = sub.SUBJECTSEQ
    INNER JOIN test t ON os.OPENEDSUBJECTSEQ = t.OPENEDSUBJECTSEQ
    INNER JOIN test_score ts ON t.TESTSEQ = ts.TESTSEQ AND st.STUDENTSEQ =
ts.STUDENTSEQ
    inner join users u on u.USERSSEQ = st.USERSSEQ;

-- 특정 과목의 특정 인원의 '평균' 계산기
-- 특정 수강생의 번호로 찾으면 됨
select studentseq as 학생번호,usersname as 학생이름,avg(TESTGRADESCORE) as 평균
from vw_student_score where studentseq = 62 GROUP BY studentseq,usersname;

-- 시험 순위 매기기
select studentseq as 학생번호,usersname as 학생이름,avg(TESTGRADESCORE) as 평균
from vw_student_score where openedsubjectseq = 2 GROUP BY studentseq,usersname
order by 평균 desc;
```

요구사항번호		작성일	2026-02-09
요구사항명	출결 현황 파악	작성자	황윤재

```
-- 뷰

CREATE OR REPLACE VIEW VW_STUDENT_ATTENDANCE_STATUS AS
SELECT
    u.usersName AS 학생이름,
    s.studentSeq AS 수강생번호,
    c.courseName AS 과정명,   -- ★ 추가된 부분: 어떤 과정인지 출력
    a.attendanceDate AS 출석일,
    at.attendanceTypeState AS 근태종류,

    -- [출석률 계산: AVG 함수 사용으로 오류 해결]
    ROUND(
```

```
    AVG(CASE WHEN at.attendanceTypeState != '결석' THEN 1 ELSE 0 END)
    OVER (PARTITION BY s.studentSeq) * 100
  , 2) || '%' AS 출석률

FROM STUDENT s
JOIN USERS u ON s.usersSeq = u.usersSeq
JOIN REGISTERED_COURSE rc ON s.registeredCourseSeq = rc.registeredCourseSeq
JOIN COURSE c ON rc.courseSeq = c.courseSeq      -- ★ 추가된 조인: 과정 테이블 연결
JOIN ATTENDANCE a ON s.studentSeq = a.studentSeq
JOIN ATTENDANCE_TYPE at ON a.attendanceTypeSeq = at.attendanceTypeSeq
WHERE s.studentStatus IN ('수강중', '수강종료');

-- 확인

SELECT * FROM VW_STUDENT_ATTENDANCE_STATUS
WHERE 수강생번호 = 93
ORDER BY 출석일 DESC;
```

C. 프로시저

1. 관리자 기능

요구사항번호		작성일	2026-02-09
요구사항명	중도 포기자 조회	작성자	김민도

```
-- procedure 중도포기
-- 목적: 출석률이 80% 이하가 되면 중도포기처리
-- 작성자: 김민도

CREATE OR REPLACE PROCEDURE proc_handle_course_drop (
   p_student_seq IN NUMBER
)
IS
```

```
    v_rg_startDate    DATE;
    v_rg_endDate      DATE;
    v_total_days      NUMBER := 0;
    v_final_absence   NUMBER := 0;
    v_attendance_rate NUMBER := 0;
    v_student_name    VARCHAR2(100);
BEGIN
    -- 1. 학생 이름과 등록된 과정 시작일, 종료일 조회
    SELECT RC.registeredCourseStartDate, RC.registeredCourseEndDate, U.usersName
    INTO v_rg_startDate, v_rg_endDate ,v_student_name
    FROM STUDENT S
    JOIN REGISTERED_COURSE RC ON S.registeredCourseSeq =
RC.registeredCourseSeq
    JOIN USERS U ON S.usersSeq = U.usersSeq
    WHERE S.studentSeq = p_student_seq;

    SELECT COUNT(*) INTO v_total_days
    FROM (
        -- 시작일(startDate)부터 종료일(endDate)까지 날짜를 한 줄씩 생성
        SELECT v_rg_startDate + LEVEL - 1 AS dt
        FROM DUAL
        CONNECT BY LEVEL <= (v_rg_endDate - v_rg_startDate + 1)
        )
        WHERE TO_CHAR(dt, 'D') NOT IN ('1', '7'); -- 1:일요일, 7:토요일 제외

    -- 2. ★미리 만든 함수 사용★ (코드 매우 깔끔!)
    v_final_absence := func_calculate_absence(p_student_seq);

    -- 3. 출석률 계산
    IF v_total_days > 0 THEN
        v_attendance_rate := ((v_total_days - v_final_absence) / v_total_days) * 100;
    ELSE
        v_attendance_rate := 0;
    END IF;
```

```
   -- 4. 80% 미만 시 상태 업데이트
   IF v_attendance_rate <= 80 THEN
      UPDATE STUDENT
      SET studentStatus = '중도포기'
      WHERE studentSeq = p_student_seq;

      DBMS_OUTPUT.PUT_LINE('[처리] ' || v_student_name || ' 학생 출석률 ' ||
ROUND(v_attendance_rate, 1) || '%로 중도포기 확정.');
   ELSE
      DBMS_OUTPUT.PUT_LINE('[처리] ' || v_student_name || ' 학생 출석률 ' ||
ROUND(v_attendance_rate, 1) || ' / ' ||'출석률 80% 이상');
   END IF;
END;
/

DECLARE
   num1 number := 1;
begin
   LOOP
      proc_handle_course_drop(num1);
      num1 := num1 + 1;
      if num1 > 80 then
         exit;
      end if;
   END LOOP;
end;
/



select * from Student s
   inner join ATTENDANCE a on a.STUDENTSEQ = s.STUDENTSEQ
where s.STUDENTSEQ = 1;
```

요구사항번호		작성일	2026-02-09
요구사항명	컴퓨터 사용 여부 업데이트	작성자	홍태훈

```
-- PROCEDURE 컴퓨터 사용 여부

CREATE OR REPLACE PROCEDURE proComputerStatus (
    p_used_status IN VARCHAR2,
    p_used_status2 IN varchar2
)
IS
BEGIN
    -- 188대의 컴퓨터 상태를 일괄 업데이트
    UPDATE COMPUTER
    SET     UsedComputer = p_used_status
    WHERE   computerSeq BETWEEN 1 AND 168;

    update computer
    set UsedComputer = p_used_status2
    where computerSeq between 169 and 188;
END;

-- 실행문
CALL proComputerStatus('사용중', '미사용');
```

요구사항번호		작성일	2026-02-09
요구사항명	훈련수당 계산	작성자	황윤재

```
-- 출석률에 따른 훈련수당 계산기
-- 년, 월을 입력받으면 전체 수강생을 돌면서 출석을 확인하고,
-- 돈을 계산해서 TRAINING_ALLOWANCE 테이블에 데이터를 작성.

-- 훈련지원금 계산 및 지급 프로시저
CREATE OR REPLACE PROCEDURE PROC_GIVE_TRAINING_ALLOWANCE (
    p_year IN NUMBER,
    p_month IN NUMBER
)
IS
    v_target_date DATE; -- 계산 기준 월 (1일)
    v_month_end DATE;   -- 계산 기준 월 (말일)

    -- 대상자 선정 커서
    CURSOR cur_student IS
        SELECT s.studentSeq, s.studentStatus
        FROM STUDENT s
        JOIN REGISTERED_COURSE rc ON s.registeredCourseSeq =
rc.registeredCourseSeq
        WHERE s.studentStatus IN ('수강중', '수강종료') -- 1. 수강대기생(X) 제외, 수료생(O)
포함
          -- 2. 기간 교차 체크 (이번 달에 수업이 있었던 사람만)
          AND rc.registeredCourseStartDate <= v_month_end
          AND rc.registeredCourseEndDate >= v_target_date;

    v_absent_count NUMBER;
    v_allowance NUMBER;
    v_dup_count NUMBER;

    v_base_pay CONSTANT NUMBER := 300000;
    v_deduction CONSTANT NUMBER := 10000;
```

```
BEGIN
   -- 날짜 세팅
   v_target_date := TO_DATE(p_year || '-' || p_month || '-01', 'YYYY-MM-DD');
   v_month_end := LAST_DAY(v_target_date);

   FOR rec IN cur_student LOOP

      -- 1. [중복 방지] 이미 이번 달 명단에 있는지 확인
      SELECT COUNT(*)
      INTO v_dup_count
      FROM TRAINING_ALLOWANCE
      WHERE studentSeq = rec.studentSeq
        AND TO_CHAR(trainingAllowanceApplyDate, 'YYYY-MM') =
TO_CHAR(v_target_date, 'YYYY-MM');

      -- 2. 명단에 없을 때만 계산 시작
      IF v_dup_count = 0 THEN

         -- A. 결석 횟수 조회 (이번 달 내역만)
         SELECT COUNT(*)
         INTO v_absent_count
         FROM ATTENDANCE a
         JOIN ATTENDANCE_TYPE atType ON a.attendanceTypeSeq =
atType.attendanceTypeSeq
         WHERE a.studentSeq = rec.studentSeq
           AND a.attendanceDate BETWEEN v_target_date AND v_month_end
           AND atType.attendanceTypeState = '결석';

         -- B. 금액 계산
         v_allowance := v_base_pay - (v_absent_count * v_deduction);
         IF v_allowance < 0 THEN v_allowance := 0; END IF;

         -- C. 데이터 입력
```

```
            INSERT INTO TRAINING_ALLOWANCE (
                trainingAllowanceSeq, trainingAllowanceAmount, trainingAllowanceApplyDate,
                trainingAllowancePayDate, trainingAllowanceState, studentSeq
            ) VALUES (
                TRAINING_ALLOWANCE_SEQ.NEXTVAL, v_allowance, SYSDATE,
                NULL, '지급예정', rec.studentSeq
            );

        END IF;

    END LOOP;

    COMMIT;
    DBMS_OUTPUT.PUT_LINE(p_year || '년 ' || p_month || '월 지급 작업 완료.');
END;
/
```

D. 트리거

1. 관리자 기능

요구사항번호		작성일	2026-02-09
요구사항명	사용자 권한 변경	작성자	박지명

```
-- 유저에서 수강생으로 insert 되면  4번권한 주기
CREATE OR REPLACE TRIGGER trg_student_insert_update_user_state
AFTER INSERT OR UPDATE ON student
FOR EACH ROW
WHEN (NEW.studentStatus = '수강중')
BEGIN
    UPDATE users
    SET authoritySeq = 4
    WHERE usersseq = :NEW.usersseq;
END;
/
insert into STUDENT (studentSeq, studentStatus, registeredCourseSeq, usersSeq) values
(student_seq.nextval, '수강중', 9, 301);

-- 수강생이 수강종료상태가 되면 유저권한이 3번으로
CREATE OR REPLACE TRIGGER trg_student_update_users_state_endcourse
AFTER UPDATE ON STUDENT
FOR EACH ROW
WHEN (new.STUDENTSTATUS = '수강종료')
BEGIN
    UPDATE USERS
    SET AUTHORITYSEQ = 3
    where USERSSEQ = :NEW.usersseq;
END;
/

-- 업데이트로 확인
update student set STUDENTSTATUS = '수강종료' where STUDENTSEQ = 301;
update student set STUDENTSTATUS = '수강중' where STUDENTSEQ = 301;
commit;
select * from users u
```

```
inner join student s on u.USERSSEQ = s.USERSSEQ where s.USERSSEQ = 301;
```

요구사항번호		작성일	2026-02-09
요구사항명	출석 반영 중도포기 계산	작성자	김민도

```
-- LECTURE TRIGGER
-- 작성자: 김민도
-- 목적: LECTURE 테이블의 lectureCapacity(정수) 컬럼을 통해 강의실에 들어올 수 있는 학생수를 트리거로 관리한다.
-- 비고:
--     1.count(*)(view에서 조회용도)를 쓰는 트리거 방식은 동시성 문제가 있음
--         > 트리거에 많은 동시 접속이 많은 환경이 예상된다면 LECTURE 테이블에 current_count같은 컬럼을 두고 관리하는 것이 좋다고 한다.
--
--     2. LECTURE - REGISTERED_COURSE - STUDENT 구조로 LECTURE와 STUDENT 테이블이 이어져 있지 않은데,
--         일반적으로 한 학생은 여러 과목을 들을 수 있고, 한 강의실에서도 시간대별로 여러 수업이 열리기 때문에,
--         만약 STUDENT에 lectureSeq를 직접 넣으면, 그 학생은 오직 한 강의실에만 소속될 수 있는 구조가 되기 쉽다.
--         현재 우리 프로젝트는 학생이 강의실을 변경하는 일이 없지만, 이 상황에도 데이터 무결성을 유지하는데 LECTURE - REGISTERED_COURSE - STUDENT 구조가 좋다.
--         학생 테이블에 강의실 번호를 직접 넣으면, 실수로 같은 수업을 듣는 학생들인데 서로 다른 강의실 번호가 입력되는 데이터 불일치 현상이 발생할 수 있기 때문이다.

CREATE OR REPLACE TRIGGER trg_check_student_limit
BEFORE INSERT ON STUDENT
FOR EACH ROW
DECLARE
   v_cap NUMBER;
   v_cur NUMBER;
```

```
BEGIN
   -- 뷰에서 현재 인원과 정원을 바로 가져옴
   SELECT lectureCapacity, currentCount
   INTO v_cap, v_cur
   FROM VIEW_COURSE_CAPACITY
   WHERE registeredCourseSeq = :NEW.registeredCourseSeq;

   -- 비교 후 에러 처리
   IF v_cur >= v_cap THEN --v_cur이 현재 인원수 이므로 참이 되면 에러를 발생시킴
      RAISE_APPLICATION_ERROR(-20001, '강의실 정원(' || v_cap || '명)이 꽉
찼습니다!');
      --호출되면 해당 SQL문은 즉시 중단(Rollback)됨 / 여러 행을 한 번에 넣는 경우 전체
작업이 취소되고 이전 상태로 되돌아감
      --이후 java에서 try-catch로 잡아서 처리하게 됨
   END IF;
END;

-- VIEW
-- 이렇게 view로 트리거의 from절을 관리하는 경우, 나중에 유지보수가 용이하고
트리거가 직관적으로 보이는 장점이 있음
CREATE OR REPLACE VIEW VIEW_COURSE_CAPACITY
AS
SELECT
   RC.registeredCourseSeq,
   L.lectureSeq,
   L.lectureCapacity,
   (SELECT COUNT(*) FROM STUDENT S WHERE S.registeredCourseSeq =
RC.registeredCourseSeq) AS currentCount
FROM REGISTERED_COURSE RC
JOIN LECTURE L ON RC.lectureSeq = L.lectureSeq;



-- VIEW와 TRIGGER 생성 후 아래 코드를 실행하면 실행되지 않는다
```

```
insert into STUDENT (studentSeq, studentStatus, registeredCourseSeq, usersSeq) values
(1, '수강완료', 1, 302);
```

요구사항번호		작성일	2026-02-09
요구사항명	사물함 상태 업데이트	작성자	홍태훈

```
CREATE OR REPLACE TRIGGER trgLockerDateStatus
AFTER INSERT OR UPDATE ON PERSONAL_LOCKER
FOR EACH ROW
DECLARE
   v_studentStatus VARCHAR2(20);
BEGIN
   -- 1. 배정받는 학생의 현재 수강 상태를 확인
   SELECT studentStatus INTO v_studentStatus
   FROM STUDENT
   WHERE studentSeq = :NEW.studentSeq;

   -- 2. 조건 체크: 학생이 '수강중'이고 + 오늘이 배정 기간(시작일~종료일) 내에 있는지
확인
   IF v_studentStatus = '수강중'
      AND SYSDATE BETWEEN :NEW.personalLockerAllocationDate AND
:NEW.personalLockerEndDate THEN

      -- 조건을 만족하면 사물함 상태를 '사용중'으로 변경
      UPDATE LOCKER
      SET lockerUseStatus = '사용중'
      WHERE lockerSeq = :NEW.lockerSeq;

   ELSE
      -- 학생이 수강생이 아니거나, 배정 기간이 아직 아니거나 이미 끝났다면 '미사용'
      UPDATE LOCKER
      SET lockerUseStatus = '미사용'
      WHERE lockerSeq = :NEW.lockerSeq;
```

```
    END IF;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        -- 학생 정보가 없을 경우 미사용 처리
        UPDATE LOCKER
        SET lockerUseStatus = '미사용'
        WHERE lockerSeq = :NEW.lockerSeq;
END;

-- 업데이트로 데이터 변경
update PERSONAL_LOCKER set personalLockerAllocationDate = '2026-01-01' where
STUDENTSEQ = 100;
update PERSONAL_LOCKER set PERSONALLOCKERENDDATE = '2026-02-28' where
STUDENTSEQ = 100;

-- 정보 확인 개인 사물함
select * from PERSONAL_LOCKER;

-- 정보 확인 사물함
select * from LOCKER;
```

요구사항번호		작성일	2026-02-09
요구사항명	컴퓨터 상태 업데이트	작성자	홍태훈

```
-- 컴퓨터 TRIGGER

CREATE OR REPLACE TRIGGER trgComputerDateStatus
AFTER INSERT OR UPDATE ON PERSONAL_COMPUTER
FOR EACH ROW
DECLARE
    v_studentStatus VARCHAR2(20);
```

```
BEGIN
   -- 1. 학생 상태 조회
   SELECT studentStatus INTO v_studentStatus
   FROM STUDENT
   WHERE studentSeq = :NEW.studentSeq;

   -- 현재(2026년)가 배정 시작일과 종료일 사이에 있는지 확인
   IF v_studentStatus = '수강중'
      AND SYSDATE BETWEEN :NEW.personalComputerAllocationDate AND
:NEW.personalComputerEndDate THEN

      UPDATE COMPUTER
      SET UsedCompuTer = '사용중'
      WHERE computerSeq = :NEW.computerSeq;

   ELSE
      UPDATE COMPUTER
      SET UsedCompuTer = '미사용'
      WHERE computerSeq = :NEW.computerSeq;
   END IF;

EXCEPTION
   WHEN NO_DATA_FOUND THEN
      UPDATE COMPUTER
      SET UsedCompuTer = '미사용'
      WHERE computerSeq = :NEW.computerSeq;
END;
/

-- 업데이트로 데이터 변경
UPDATE PERSONAL_COMPUTER SET personalComputerAllocationDate = '2026-01-02'
WHERE STUDENTSEQ = 100;
update PERSONAL_COMPUTER set personalComputerEndDate = '2026-02-01' where
STUDENTSEQ = 100;
```

```
-- 정보 확인 개인 컴퓨터
select * from PERSONAL_COMPUTER;

-- 정보 확인 컴퓨터
select * from COMPUTER;
```

2. 교사 기능

요구사항번호		작성일	2026-02-09
요구사항명	배점 유효성 검사	작성자	황윤재

```
-- 배점 유효성 검사 트리거
-- 배점의 총합은 정확히 100이어야 함

CREATE OR REPLACE TRIGGER TRG_CHECK_POINT_SUM
BEFORE INSERT OR UPDATE ON POINT_VALUE
FOR EACH ROW
DECLARE
   v_total_sum NUMBER;
BEGIN
   -- 1. 입력되는 세 가지 배점의 합계를 계산
   v_total_sum := :NEW.pointValueAttendance
              + :NEW.pointValueWrittenTest
              + :NEW.pointValuePracticalTest;

   -- 2. 배점의 합계가 '정확히 100'이 아니면 에러 발생
   IF v_total_sum != 100 THEN
      RAISE_APPLICATION_ERROR(-20002,
         '배점 비율의 총합은 정확히 100이어야 합니다. (현재 입력된 합계: ' || v_total_sum
|| '점)');
   END IF;
END;
```

```
/

-- 합계: 50 + 50 + 10 = 110 (실패 -> 트리거 에러 발생)
/*
INSERT INTO POINT_VALUE (pointValueSeq, pointValueAttendance,
pointValueWrittenTest, pointValuePracticalTest, openedSubjectSeq)
VALUES (POINT_VALUE_SEQ.NEXTVAL, 50, 50, 10, 78);
*/
```

요구사항번호		작성일	2026-02-09
요구사항명	시험 점수 유효성 검사	작성자	황윤재

```
-- test_score 테이블에 insert 나 update 될 때
-- 점수가 0 미만이거나 100 초과면 에러를 발생시키는
-- 시험 점수 유효성 검사 트리거

CREATE OR REPLACE TRIGGER TRG_CHECK_VALID_SCORE
BEFORE INSERT OR UPDATE ON TEST_SCORE
FOR EACH ROW
BEGIN
   -- 들어오는 점수(:NEW.testGradeScore)가 0보다 작거나 100보다 크면 에러 발생
   IF :NEW.testGradeScore < 0 OR :NEW.testGradeScore > 100 THEN
      RAISE_APPLICATION_ERROR(-20001, '시험 점수는 0점 이상 100점 이하이어야
합니다.');
   END IF;
END;
/

-- 1. 실패 케이스 (100점 초과) -> 에러 발생해야 함!
/*
INSERT INTO TEST_SCORE (testGradeSeq, testGradeScore, testGradeParticipated,
testSeq, studentSeq)
```

```
VALUES (TEST_SCORE_SEQ.NEXTVAL, 150, '참여', 1, 1);
*/
-- 결과: ORA-20001: 시험 점수는 0점 이상 100점 이하이어야 합니다.
-- 2. 실패 케이스 (음수) -> 에러 발생해야 함!
/*
INSERT INTO TEST_SCORE (testGradeSeq, testGradeScore, testGradeParticipated,
testSeq, studentSeq)
VALUES (TEST_SCORE_SEQ.NEXTVAL, -10, '참여', 1, 1);
*/
-- 결과: ORA-20001: 시험 점수는 0점 이상 100점 이하이어야 합니다.
```