**DB Object 스크립트(PL-SQL)**

|  |  |  |  |
| --- | --- | --- | --- |
| 요구사항번호 | C-01 | 작성일 | 2025-02-06 |
| 요구사항명 | 관리자 계정 요구사항 | 작성자 | 4조(정다음) |
| DML | | | |
| CREATE OR REPLACE PROCEDURE managerAccount (  p\_password IN mManager.password%TYPE  ) IS  v\_manager\_idmManager.manager\_id%TYPE;  v\_namemManager.name%TYPE;  v\_passwordmManager.password%TYPE;  BEGIN  SELECT manager\_id, name, password  INTO v\_manager\_id, v\_name, v\_password  FROM mManager  WHERE password = p\_password;  DBMS\_OUTPUT.PUT\_LINE('관리자 번호: ' || v\_manager\_id || ', 관리자 이름: ' || v\_name);  DBMS\_OUTPUT.PUT\_LINE('로그인 되었습니다.');  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('잘못된 비밀번호입니다.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('오류가 발생했습니다: ' || SQLERRM);  END;  /  begin  managerAccount('222221');  end;  / | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 요구사항번호 | C-04 | 작성일 | 2025-02-06 |
| 요구사항명 | 모든 개설 과정 등록 및 관리 가능 | 작성자 | 4조(정다음) |
| DML | | | |
| -- 과정 생성, 수정시 트리거  CREATE OR REPLACE TRIGGER trg\_mopencurriculum\_validation  BEFORE INSERT OR UPDATE ON mopencurriculum  FOR EACH ROW  DECLARE  v\_count NUMBER;  BEGIN  SELECT COUNT(\*) INTO v\_count FROM mcurriculum WHERE curriculum\_id = :NEW.curriculum\_id;  IF v\_count = 0 THEN  RAISE\_APPLICATION\_ERROR(-20001, '존재하지 않는 과정입니다.');  END IF;  SELECT COUNT(\*) INTO v\_count FROM mRoom WHERE room\_id = :NEW.room\_id;  IF v\_count = 0 THEN  RAISE\_APPLICATION\_ERROR(-20002, '잘못된 강의실 번호입니다.');  END IF;  SELECT COUNT(\*) INTO v\_count FROM mteacher WHERE teacher\_id = :NEW.teacher\_id;  IF v\_count = 0 THEN  RAISE\_APPLICATION\_ERROR(-20003, '잘못된 교사 번호입니다.');  END IF;  END;  / | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 요구사항번호 | C-06 | 작성일 | 2025-02-06 |
| 요구사항명 | 모든 교육생 정보 등록 및 관리 | 작성자 | 4조(정다음) |
| DML | | | |
| /\*교육생 정보 입력 시>교육생 이름, 주민번호 뒷자리, 전화번호를 기본 입력, 등록일 자동 입력\*/  CREATE OR REPLACE PROCEDURE insert\_student(  p\_name IN VARCHAR2,  p\_jumin IN VARCHAR2,  p\_tel IN VARCHAR2  )  IS  v\_id NUMBER;  BEGIN  SELECT COUNT(\*) + 1 INTO v\_id FROM mStudent;  INSERT INTO mStudent (student\_id, name, jumin, tel, registration\_date)  VALUES (v\_id, p\_name, p\_jumin, p\_tel, SYSDATE);  dbms\_output.put\_line(p\_name || ' 교육생이 등록되었습니다.');  COMMIT;  END insert\_student;  /  begin  insert\_student('조성제', '1822113', '010-6822-3729');  end;  /  /\*교육생 검색> 날짜 데이터 입력\*/  CREATE INDEX idx\_student\_name\_jumin ON vwStudentAlldate(교육생명, "주민번호 뒷자리"); | | | |
| CREATE OR REPLACE PROCEDURE studentsearch(  p\_name IN vwStudentAlldate.교육생명%TYPE,  p\_jumin IN vwStudentAlldate."주민번호 뒷자리"%TYPE  )  IS  v\_namevwStudentAlldate.교육생명%TYPE;  v\_juminvwStudentAlldate."주민번호 뒷자리"%TYPE;  v\_currinamevwStudentAlldate."개설 과정명"%TYPE;  v\_statusvwStudentAlldate."개설 현황"%TYPE;  CURSOR c1 IS  SELECT 교육생명, "주민번호 뒷자리", "개설 과정명", "개설 현황"  FROM vwStudentAlldate  WHERE 교육생명 = p\_name  AND "주민번호 뒷자리" = p\_jumin;  BEGIN  OPEN c1;  LOOP  FETCH c1 INTO v\_name, v\_jumin, v\_curriname, v\_status;  EXIT WHEN c1%NOTFOUND;  DBMS\_OUTPUT.PUT\_LINE('교육생명: ' || v\_name || ', 주민번호 뒷자리: ' || v\_jumin || ', 개설 과정명: ' || v\_curriname || ', 개설 현황: ' || v\_status);  END LOOP;  CLOSE c1;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('해당 교육생 정보가 없습니다.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('오류가 발생했습니다: ' || SQLERRM);  end studentsearch;  /  begin  studentsearch('김희재','1065487');  end;  / | | | |
| /\* 교육생에 대한 수료 및 중도 탈락 처리 필요 > 날짜 데이터 입력\*/  select \* from mStudent;  create or replace PROCEDURE studentdropdate(  p\_name IN mStudent.name%TYPE,  p\_jumin IN mStudent.jumin%TYPE,  p\_dropdate IN mStudent.dropout\_date%TYPE  )  is  v\_namemStudent.name%type;  v\_juminmStudent.jumin%type;  v\_telmStudent.tel%type;  v\_regdatemStudent.registration\_date%type;  v\_dropdatemStudent.dropout\_date%type;  begin  begin  select name, jumin, tel, registration\_date, dropout\_date  into v\_name, v\_jumin, v\_tel, v\_regdate, v\_dropdate  from mStudent  where jumin = p\_jumin;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('해당 교육생이 존재하지 않습니다.');  RETURN;  end;  IF v\_dropdate IS NOT NULL THEN  DBMS\_OUTPUT.PUT\_LINE('해당 날짜로 중퇴 처리 된 교육생입니다. 중퇴일: ' || v\_dropdate);  RETURN;  END IF; | | | |
| update mStudent set dropout\_date = p\_dropdate where jumin = p\_jumin;  DBMS\_OUTPUT.PUT\_LINE('교육생명: ' || v\_name || ', 주민번호 뒷자리: ' || v\_jumin || ', 연락처: ' || v\_tel || ', 등록일: ' || v\_regdate);  DBMS\_OUTPUT.PUT\_LINE(v\_dropdate || '로 중퇴일 등록하였습니다.');  end;  /  begin  studentdropdate('김희재', '1065487', '2030-01-01');  end;  / | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 요구사항번호 | C-07 | 작성일 | 2025-02-06 |
| 요구사항명 | 수강생 기준 필기, 실기 점수 실데이터를 null 로 변경하는 프로시저 | 작성자 | 조인제 |
| DML | | | |
| -- 중도탈락된 수강생 기준 필기, 실기 점수 실데이터를 null 로 변경하는 프로시저 작성하기  CREATE OR REPLACE PROCEDURE DropoutStudentScoreUpdate  IS  BEGIN  -- 중도탈락된 교육생이 dropout\_date 이후에 본 시험 점수를 NULL로 업데이트  UPDATE mExamResulter  SET er.score = NULL  WHERE EXISTS (  SELECT 1  FROM mStudent s  JOIN mExam e ON er.exam\_id = e.exam\_id  WHERE er.student\_id = s.student\_id  AND s.dropout\_date IS NOT NULL  AND e.exam\_date>s.dropout\_date -- dropout\_date 이후의 시험만 반영  );  COMMIT;  END;  /  begin  DropoutStudentScoreUpdate;  end;  /  -- 중도탈락생 시험 점수 조회  SELECT er.student\_id, er.exam\_id, er.score, s.dropout\_date, e.exam\_date  FROM mExamResulter  JOIN mStudent s ON er.student\_id = s.student\_id  JOIN mExam e ON er.exam\_id = e.exam\_id  WHERE s.dropout\_date IS NOT NULL  AND e.exam\_date>s.dropout\_date; inner join tblapplicant app  on app.applicant\_seq = int.applicant\_seq  inner join tblcurriculumcul  on cul.curriculum\_seq = app.curriculum\_seq  inner join tbllecturelec  on lec.curriculum\_seq = cul.curriculum\_seq  where substr(to\_char(app.applicantssn),8) = '1448626'; -- 교육생 본인 주민등록번호 뒷자리 7글자 입력 | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 요구사항번호 | D-01 | 작성일 | 2025-02-06 |
| 요구사항명 | 교사 로그인 프로시저 | 작성자 | 4조(정다음) |
| DML | | | |
| CREATE OR REPLACE PROCEDURE teacherAccount (  p\_jumin IN mTeacher.jumin%TYPE  ) AS  v\_teacher\_idmTeacher.teacher\_id%TYPE;  v\_namemTeacher.name%TYPE;  v\_telmTeacher.tel%TYPE;  v\_juminmTeacher.jumin%TYPE;  BEGIN  SELECT teacher\_id, name, tel, jumin  INTO v\_teacher\_id, v\_name, v\_tel, v\_jumin  FROM mTeacher  WHERE jumin = p\_jumin;  DBMS\_OUTPUT.PUT\_LINE('교사 번호: ' || v\_teacher\_id || ', 이름: ' || v\_name || ', 전화번호: ' || v\_tel);  DBMS\_OUTPUT.PUT\_LINE(v\_name||'선생님, 로그인 되었습니다.');  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('잘못된 비밀번호입니다.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('오류가 발생했습니다: ' || SQLERRM);  END teacherAccount;  /  begin  teacherAccount('1645784');  end;  / | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 요구사항번호 | D-03 | 작성일 | 2025-02-06 |
| 요구사항명 | 배점입출력 | 작성자 | 김형년 |
| DML | | | |
| **- 배점등록이안된경우과목정보출력시 null로표시**  --시험문제생성시배점 null로자동생성  CREATE OR REPLACE TRIGGER trgDistribution  AFTER  INSERT  ON mExam  FOR EACH ROW  BEGIN  INSERT INTO distributionOfScores(distribution\_id,  attendance,  written,  practical,  exam\_id)  VALUES((SELECT COUNT(\*) FROM distributionOfScores),  null,  null,  null,  :new.exam\_id);  END;  / | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 요구사항번호 | E-04 | 작성일 | 2025-02-06 |
| 요구사항명 | 강의실예약 | 작성자 | 김형년 |
| DML | | | |
| CREATE OR REPLACE PROCEDURE procReservationCreate(  ptel IN VARCHAR2,  proom\_id IN NUMBER,  preservationDate IN DATE,  preservationTime IN DATE,  ppersonnel IN NUMBER  )  IS  vcountTel number;  vcheckDatemHoliday.dateclassfication%type;  vcheckHour number;  vcountSame number;  vnamemReservation.name%type  BEGIN  select count(\*)  INTO vcountTel  from mStudent  where tel = ptel;  IF vcountTel = 0 then  DBMS\_OUTPUT.PUT\_LINE('등록된교육생이아닙니다');  RETURN;  END IF;  -- **당일부터 + 7일까지만예약가능..**  **-- 예약이불가능하면오류메시지반환.**  IF preservationDate<sysdate OR preservationDate> (sysdate +7) THEN  DBMS\_OUTPUT.PUT\_LINE('예약가능한날짜가아닙니다');  RETURN;  END IF;  SELECT dateClassfication  INTO vcheckDate  FROM mHoliday  WHERE preservationDate = totalDate;  SELECT TO\_NUMBER(TO\_CHAR(preservationTime, 'HH24'))  INTO vcheckHour  FROM dual;  **-- 예약공휴일에는불가**  IF vcheckDate = '공휴일' THEN -- 공휴일확인  DBMS\_OUTPUT.PUT\_LINE('공휴일에는예약이불가능합니다.');  RETURN;  ELSIF vcheckDate = '주말' THEN -- 주말확인  IF vcheckHour< 09 and vcheckHour> 18 THEN  DBMS\_OUTPUT.PUT\_LINE('주말에는 09:00 ~ 18:00에만이용가능합니다.');  RETURN;  END IF;  ELSIF vcheckDate = '평일' THEN -- 평일확인  IF vcheckHour< 18 and vcheckHour> 22 THEN  DBMS\_OUTPUT.PUT\_LINE('평일에는 18:00 ~ 22:00에만이용가능합니다.');  RETURN;  END IF;  END IF;  SELECT name  INTO vname  FROM mStudent  WHERE tel = ptel;  SELECT count(\*)  INTO vcountSame  FROM mReservation  WHERE proom\_id = room\_id and  preservationDate = reservationDate and  **-- 예약시간대는한시간단위.**  TO\_CHAR(preservationTime, 'HH24:MI')  || ' ~ ' ||  TO\_CHAR(preservationTime + INTERVAL '1' HOUR, 'HH24:MI') = reservationTime;  **-- 예약자가입력한시간대와강의실의충돌을확인**  **-- 예약하려는강의실의예약가능여부를확인**  IF vcountSame> 0 THEN  DBMS\_OUTPUT.PUT\_LINE('예약할수없습니다.');  DBMS\_OUTPUT.PUT\_LINE('다른시간대를선택해주세요.');  RETURN;  END IF;  **-- 예약이가능하다면, 해당강의실을예약처리**  INSERT INTO mReservation(reservationNumber, name, tel, room\_id, reservationDate, reservationTime, personnel)  VALUES((select count(\*) from mReservation) + 1,  vname,  ptel,  proom\_id,  preservationDate,  TO\_CHAR(preservationTime, 'HH24:MI')  || ' ~ ' ||  TO\_CHAR(preservationTime + INTERVAL '1' HOUR, 'HH24:MI'),  ppersonnel);  commit;  DBMS\_OUTPUT.PUT\_LINE('예약이완료되었습니다');  END procReservationCreate;  /  BEGIN  procReservationCreate(  '010-3737-2375',  6,  TO\_DATE('2025-02-13', 'YYYY-MM-DD'),  TO\_DATE('19:00', 'HH24:MI'),  4  );  END;  / | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 요구사항번호 | E-01 | 작성일 | 2025-02-06 |
| 요구사항명 | 교육생 계정 | 작성자 | 김두현 |
| DML | | | |
| CREATE OR REPLACE PROCEDURE proc\_studentLogin (  p\_jumin IN mStudent.jumin%TYPE  ) AS  v\_student\_idmStudent.student\_id%TYPE;  v\_namemStudent.name%TYPE;  v\_telmStudent.tel%TYPE;  v\_juminmStudent.jumin%TYPE;  BEGIN  SELECT student\_id, name, tel, jumin  INTO v\_student\_id, v\_name, v\_tel, v\_jumin  FROM mStudent  WHERE jumin = p\_jumin;  DBMS\_OUTPUT.PUT\_LINE(' 교육생번호: ' || v\_student\_id || ', 이름: ' || v\_name || ', 전화번호: ' || v\_tel);  DBMS\_OUTPUT.PUT\_LINE(v\_name||'교육생님, 로그인 되었습니다.');  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  -- 비밀번호가 틀린 경우  DBMS\_OUTPUT.PUT\_LINE('잘못된 비밀번호입니다.');  WHEN OTHERS THEN  -- 다른 예외 처리  DBMS\_OUTPUT.PUT\_LINE('오류가 발생했습니다: ' || SQLERRM);  END studentAccount;  /  begin  studentAccount('4722631');  end;  /  -- 로그인 과정 통과 시 -> 교육생 개인 정보출력  select \* from vwStudentInfo;  -- 성적 정보는 과목별 목록 형태로 출력  select \* from vwStudentResult;  -- 출결 관리 및 출결 조회  ALTER SESSION SET NLS\_DATE\_FORMAT = 'YYYY-MM-DD HH24:MI:SS'; -- 출결정보 시 필요  select \* from vwStudentAttendance;  ALTER SESSION SET NLS\_DATE\_FORMAT = 'YYYY-MM-DD'; -- 출결 정보 출력 후 실행(날짜 set up reset) | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 요구사항번호 | E-03 | 작성일 | 2025-02-06 |
| 요구사항명 | 출결 관리 및 출결 조회 | 작성자 | 김두현 |
| DML | | | |
| CREATE OR REPLACE PROCEDURE proc\_timekeeping (  p\_student\_id NUMBER,  p\_mode VARCHAR2, -- 'IN(입실)', 'OUT(퇴실)', 'LEAVE(외출)', 'SICK(병가)', 'ETC(기타)'  p\_attendance\_type\_id NUMBER DEFAULT NULL -- 수동 입력 시 사용할 attendance\_type\_id  )  IS  v\_count NUMBER;  v\_new\_id NUMBER;  v\_attendance\_type\_id NUMBER;  v\_clock\_in DATE;  v\_clock\_out DATE;  v\_student\_name VARCHAR2(1500);  v\_existing\_attendance\_type\_id NUMBER; -- 기존 출결 유형 저장  BEGIN  -- 교육생 이름 조회  SELECT name INTO v\_student\_name  FROM mStudent  WHERE student\_id = p\_student\_id;  IF p\_mode = 'IN' THEN  -- 기존에 같은 날 입실 기록이 있는지 확인 (유효성 체크)  SELECT COUNT(\*)  INTO v\_count  FROM mTimeKeeping  WHERE student\_id = p\_student\_id  AND TRUNC(clock\_in) = TRUNC(SYSDATE);  IF v\_count = 0 THEN  SELECT NVL(MAX(timekeeping\_id), 0) + 1 INTO v\_new\_id FROM mTimeKeeping;  -- 입실시간 09:00을 기준으로 지각, 정상 구분  IF TO\_CHAR(SYSDATE, 'HH24:MI:SS') > '09:00:00' THEN  v\_attendance\_type\_id := 2; -- 지각  ELSE  v\_attendance\_type\_id := 1; -- 정상  END IF;  -- 입실 기록 추가  INSERT INTO mTimeKeeping (timekeeping\_id, clock\_in, student\_id, attendance\_type\_id)  VALUES (v\_new\_id, SYSDATE, p\_student\_id, v\_attendance\_type\_id);  DBMS\_OUTPUT.PUT\_LINE('안녕하세요.' || v\_student\_name || ' 교육생님, 정상 입실 처리 되었습니다.');  ELSE  DBMS\_OUTPUT.PUT\_LINE(v\_student\_name || ' 교육생님, 이미 입실 기록이 있습니다.');  END IF;  ELSIF p\_mode = 'OUT' THEN  -- 가장 최근 입실 기록 찾기  SELECT clock\_in, attendance\_type\_id INTO v\_clock\_in, v\_existing\_attendance\_type\_id  FROM mTimeKeeping  WHERE student\_id = p\_student\_id AND clock\_out IS NULL  ORDER BY clock\_in DESC  FETCH FIRST 1 ROW ONLY;  IF v\_clock\_in IS NOT NULL THEN  -- 퇴실 시간 기록  UPDATE mTimeKeeping  SET clock\_out = SYSDATE  WHERE student\_id = p\_student\_id AND clock\_out IS NULL;  -- 조퇴 여부 판별  IF TO\_CHAR(SYSDATE, 'HH24:MI:SS') < '17:50:00' THEN  v\_attendance\_type\_id := 3; -- 조퇴  ELSE  v\_attendance\_type\_id := v\_existing\_attendance\_type\_id; -- 정상 퇴실은 기존 출결 상태 유지  END IF;  -- 출결 유형 업데이트  UPDATE mTimeKeeping  SET attendance\_type\_id = v\_attendance\_type\_id  WHERE student\_id = p\_student\_id AND clock\_out = SYSDATE;  DBMS\_OUTPUT.PUT\_LINE('안녕하세요.' || v\_student\_name || ' 교육생님, 정상 퇴실 처리 되었습니다.');  ELSE  DBMS\_OUTPUT.PUT\_LINE(v\_student\_name || ' 교육생님, 입실 기록이 없어 퇴실할 수 없습니다.');  END IF;  -- 수동 입력 (외출, 병가, 기타)  ELSIF p\_mode IN ('LEAVE', 'SICK', 'ETC') THEN  SELECT COUNT(\*)  INTO v\_count  FROM mTimeKeeping  WHERE student\_id = p\_student\_id  AND attendance\_type\_id IN (4, 5, 6) -- 외출, 병가, 기타  AND TRUNC(clock\_in) = TRUNC(SYSDATE); -- 같은 날에 이미 기록이 있는지 확인  IF v\_count = 0 THEN  -- 수동 출결 유형 설정 (외출, 병가, 기타)  IF p\_attendance\_type\_id IS NULL THEN  DBMS\_OUTPUT.PUT\_LINE('출결 유형번호를 입력하세요. ("외출" = 4, "병가" = 5, "기타" = 6)');  RETURN;  END IF;  -- 새로운 timekeeping\_id 생성  SELECT NVL(MAX(timekeeping\_id), 0) + 1 INTO v\_new\_id FROM mTimeKeeping;  -- 출결 기록 추가 (clock\_in, clock\_out은 NULL)  INSERT INTO mTimeKeeping (timekeeping\_id, clock\_in, clock\_out, student\_id, attendance\_type\_id)  VALUES (v\_new\_id, NULL, NULL, p\_student\_id, p\_attendance\_type\_id);  DBMS\_OUTPUT.PUT\_LINE(v\_student\_name || ' 교육생님, ' || p\_mode || ' 처리 완료');  ELSE  DBMS\_OUTPUT.PUT\_LINE(v\_student\_name || ' 교육생님, 오늘은 이미 ' || p\_mode || ' 기록이 있습니다.');  END IF;  ELSE  DBMS\_OUTPUT.PUT\_LINE('잘못된 입력입니다. "IN", "OUT", "LEAVE", "SICK", "ETC" 중 하나를 입력하세요.');  END IF;  COMMIT;  END;  /  rollback;  SET SERVEROUTPUT ON;  -- 입실  EXEC proc\_timekeeping(285, 'IN');  -- 퇴실  EXEC proc\_timekeeping(285, 'OUT');  --수동입력  -- 외출  EXEC proc\_timekeeping(285, 'LEAVE', 4);  -- 병가  EXEC proc\_timekeeping(285, 'SICK', 5);  -- 기타  EXEC proc\_timekeeping(285, 'ETC', 6);  -- 출결 데이터 확인  SELECT \* FROM mTimeKeeping WHERE student\_id = 285;  DELETE FROM mTimeKeeping WHERE student\_id = 285;  drop procedure proc\_timekeeping;  ALTER SESSION SET NLS\_DATE\_FORMAT = 'YYYY-MM-DD HH24:MI:SS'; -- 출결정보 시 필요  ALTER SESSION SET NLS\_DATE\_FORMAT = 'YYYY-MM-DD'; -- 출결 정보 출력 후 실행(날짜 set up reset) | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 요구사항번호 | F-01 | 작성일 | 2025-02-06 |
| 요구사항명 | 수료생 사후관리 | 작성자 | 김두현 |
| DML | | | |
| -- 1-1. 전체 수료생 평균연봉 조회  select \* from mGradStudent;  select \* from mAfterService order by afterService\_id;  SET SERVEROUTPUT ON;  CREATE OR REPLACE PROCEDURE gradAvgSalary (  p\_avg\_salary OUT number  )  AS  BEGIN  select TRUNC(AVG(annualSalary)) into p\_avg\_salary  from mAfterService;  DBMS\_OUTPUT.PUT\_LINE(' 전체 교육생 평균 연봉: ' || p\_avg\_salary || '만원');  IF p\_avg\_salary IS NULL THEN p\_avg\_salary := 0;  END IF;  EXCEPTION  WHEN OTHERS THEN  -- 다른 예외 처리  DBMS\_OUTPUT.PUT\_LINE('오류가 발생했습니다: ' || SQLERRM);  END gradAvgSalary;  /  DECLARE  v\_avg\_salaryNUMBER; -- OUT 파라미터의 결과를 받을 변수 선언  BEGIN  gradAvgSalary(v\_avg\_salary);  END;  /  -- 1-2. 전체 수료생 연도별 평균연봉 조회  CREATE OR REPLACE PROCEDURE p\_avg\_yearly\_salary (  p\_year IN NUMBER, -- 입력: 조회할 연도  p\_avg\_salary OUT NUMBER -- 출력: 평균 연봉  )  IS  BEGIN  SELECT AVG(annualSalary) INTO p\_avg\_salary  FROM mAfterService  WHERE EXTRACT(YEAR FROM enterDate) = p\_year;  DBMS\_OUTPUT.PUT\_LINE(p\_year||'년 입사자 평균 연봉: '||TO\_CHAR(p\_avg\_salary, '999,999,999')||'만원');  IF p\_avg\_salary IS NULL THEN  p\_avg\_salary := 0;  END IF;  END p\_avg\_yearly\_salary;  /  DECLARE  v\_avg\_salary NUMBER;  BEGIN  p\_avg\_yearly\_salary(2024, v\_avg\_salary);  END;  /  -- 2.1 수료생 수료후 입사일까지 평균 소요일  CREATE OR REPLACE PROCEDURE AfterServiceAvgEnterDate (  p\_avg\_enter\_date OUT NUMBER  )  AS  BEGIN  select TRUNC(AVG(enterdate - finishdate)) into p\_avg\_enter\_date  from mAfterService;  DBMS\_OUTPUT.PUT\_LINE('전체 교육생 평균 입사 소요일: ' || p\_avg\_enter\_date || '일');  IF p\_avg\_enter\_date IS NULL THEN p\_avg\_enter\_date := 0;  END IF;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('오류가 발생했습니다: ' || SQLERRM);  END AfterServiceAvgEnterDate;  /  DECLARE  p\_avg\_enter\_date NUMBER;  BEGIN  AfterServiceAvgEnterDate(p\_avg\_enter\_date);  END;  /  --2.2 연도별 수료생 수료후 입사일까지 평균 소요일  CREATE OR REPLACE PROCEDURE AfterServiceYearlyAvgEnterDate (  p\_year IN NUMBER,  p\_yearly\_avg\_enter\_date OUT NUMBER  )  AS  BEGIN  SELECT ROUND(AVG(enterdate - finishdate)) INTO p\_yearly\_avg\_enter\_date  FROM mAfterService  WHERE EXTRACT(YEAR FROM enterDate) = p\_year;  DBMS\_OUTPUT.PUT\_LINE('연도별 교육생 평균 입사 소요일: ' || p\_yearly\_avg\_enter\_date || '일');  IF p\_yearly\_avg\_enter\_date IS NULL THEN p\_yearly\_avg\_enter\_date := 0;  END IF;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('오류가 발생했습니다: ' || SQLERRM);  END AfterServiceYearlyAvgEnterDate;  /  DECLARE  p\_yearly\_avg\_enter\_date NUMBER;  BEGIN  AfterServiceYearlyAvgEnterDate(2024, p\_yearly\_avg\_enter\_date);  END;  / | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 요구사항번호 | F-03 | 작성일 | 2025-02-06 |
| 요구사항명 | 자료 관리 및 자료 조회 요구사항 | 작성자 | 4조(정다음) |
| DML | | | |
| CREATE OR REPLACE TRIGGER mmaterial\_validation  BEFORE INSERT OR UPDATE OR DELETE ON mmaterial  FOR EACH ROW  DECLARE  v\_count NUMBER;  BEGIN  -- mteacher와 mopencurriculum의 관계 검증 (vwMaterial 활용)  SELECT COUNT(\*) INTO v\_count FROM vwMaterial  WHERE teacher\_id= :NEW.teacher\_id AND open\_curriculum\_id = :NEW.open\_curriculum\_id;  IF v\_count = 0 THEN  RAISE\_APPLICATION\_ERROR(-20004, '현재 교사가 진행하는 과정이 아닙니다.');  END IF;  END;  /  CREATE OR REPLACE TRIGGER mmaterial\_validation  BEFORE INSERT OR UPDATE OR DELETE ON mmaterial  FOR EACH ROW  DECLARE  v\_count NUMBER;  BEGIN  -- mteacher와 mopencurriculum의 관계 검증 (vwMaterial 활용)  SELECT COUNT(\*) INTO v\_count FROM vwMaterial  WHERE teacher\_id= :NEW.teacher\_id AND open\_curriculum\_id = :NEW.open\_curriculum\_id;  IF v\_count = 0 THEN  RAISE\_APPLICATION\_ERROR(-20004, '현재 교사가 진행하는 과정이 아닙니다.');  END IF;  END;  /  --강의실 추가  create or replace procedure procClassAdd(  pClassNum in tblclass.classnum%type,  pClassLimit in tblclass.classlimit%type  )  is  begin  insert into tblClass(ClassNum,ClassLimit) values(pClassNum, pClassLimit);  commit;  dbms\_output.put\_line('INSERT SUCCESS');  end;  --강의실 삭제  create or replace procedure procClassDel(  pClassNum in tblClass.ClassNum%type  )  is  begin  delete from tblClass  where pClassNum = ClassNum;  commit;  end;  create or replace procedure procClassUpdate(  pClassNum in tblclass.classnum%type,  pClassLimit in tblclass.classlimit%type  )  is  begin  update tblClass set classlimit = pClassLimit where classNum = pClassnum;  commit;  dbms\_output.put\_line('INSERT SUCCESS');  end; | | | |