**오브젝트 스크립트(PLSQL)**

**관리자**

2조

한가람, 김원중, 이승진, 이지온, 이지원

|  |  |  |  |
| --- | --- | --- | --- |
| 요구사항번호 | A-1.관리자 계정 | 작성일 | 2025-02-07 |
| 요구사항명 | 관리자계정 | 작성자 | 김원중 |
| DML | | | |
| /\* 관리자 로그인 \*/  CREATE OR REPLACE PROCEDURE procLoginAdmin(  p\_AdminmanagerId IN NUMBER,  p\_AdminPassword IN VARCHAR2  ) AS  v\_countNUMBER;  -- 로그인 시 출력할 데이터를 담을 변수들  v\_AdminNametblAdmin.AdminName%type;  v\_AdminmanagerIdtblAdmin.AdminmanagerId%type;  BEGIN  -- 해당 관리자 존재하는지 확인  SELECT COUNT(\*) INTO v\_count  FROM tblAdmin  WHERE AdminmanagerId = p\_AdminmanagerId  AND ADMINPASSWORD = p\_ADMINPASSWORD;  -- 로그인 결과 처리  IF v\_count> 0 THEN  DBMS\_OUTPUT.PUT\_LINE('');  DBMS\_OUTPUT.PUT\_LINE('=================================');  DBMS\_OUTPUT.PUT\_LINE(' [쌍용 교육센터] ');  DBMS\_OUTPUT.PUT\_LINE('=================================');  DBMS\_OUTPUT.PUT\_LINE(' 관리자 로그인 성공 ');  DBMS\_OUTPUT.PUT\_LINE('---------------------------------');  -- 관리자 정보 조회  SELECT  tblAdmin.AdminName,  tblAdmin.AdminmanagerId  INTO  v\_AdminName, v\_AdminmanagerId  FROM tblAdmin  WHERE tblAdmin.AdminmanagerId = p\_AdminmanagerId;  DBMS\_OUTPUT.PUT\_LINE('이름 : ' || v\_AdminName);  DBMS\_OUTPUT.PUT\_LINE('관리자 ID : ' || v\_AdminmanagerId);  DBMS\_OUTPUT.PUT\_LINE('=================================');  DBMS\_OUTPUT.PUT\_LINE('');  ELSE  DBMS\_OUTPUT.PUT\_LINE('로그인 실패');  END IF;  END procLoginAdmin;  /  -- 관리자 로그인 검증  EXEC procLoginAdmin(1, '2158173'); | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 요구사항번호 | A-2.기초 정보 관리 기능 | 작성일 | 2025-02-07 |
| 요구사항명 | 기초 정보 관리 | 작성자 | 김원중 |
| DML | | | |
| /\* 강의과정 관리 \*/  --1. 강의 과정 등록  CREATE OR REPLACE PROCEDURE procInsertCourse(  p\_courseName IN tblCourse.courseName%TYPE,  p\_courseDuration IN tblCourse.courseDuration%TYPE,  p\_courseObjective IN tblCourse.courseObjective%TYPE  ) AS  v\_newCourseIdNUMBER; -- 새롭게 생성될 과정 ID  BEGIN  -- 새로운 과정 ID 생성 (현재 최대 courseId + 1)  SELECT NVL(MAX(courseId), 0) + 1 INTO v\_newCourseId FROM tblCourse;  -- 과정 정보 삽입  INSERT INTO tblCourse(courseId, courseName, courseDuration, courseObjective)  VALUES (v\_newCourseId, p\_courseName, p\_courseDuration, p\_courseObjective);  -- 등록 성공 메시지 출력  DBMS\_OUTPUT.PUT\_LINE('');  DBMS\_OUTPUT.PUT\_LINE('등록 성공!');  DBMS\_OUTPUT.PUT\_LINE('');  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('');  DBMS\_OUTPUT.PUT\_LINE('개설과정 코드 : ' || v\_newCourseId);  DBMS\_OUTPUT.PUT\_LINE('개설 과정명 : ' || p\_courseName);  DBMS\_OUTPUT.PUT\_LINE('개설 과정 기간 : ' || p\_courseDuration || '개월');  DBMS\_OUTPUT.PUT\_LINE('개설 과정 목표 : ' || p\_courseObjective);  DBMS\_OUTPUT.PUT\_LINE('');  DBMS\_OUTPUT.PUT\_LINE('======================================================');  -- 커밋 수행  COMMIT;  EXCEPTION  WHEN OTHERS THEN  -- 오류 발생 시 예외 처리  DBMS\_OUTPUT.PUT\_LINE('등록 실패, 정보를 다시 확인해주세요!');  ROLLBACK;  END procInsertCourse;  /  --2. 강의 과정 삭제  CREATE OR REPLACE PROCEDURE procDeleteCourse(  p\_courseId IN tblCourse.courseId%TYPE  ) AS  v\_courseNametblCourse.courseName%TYPE;  v\_courseDurationtblCourse.courseDuration%TYPE;  v\_courseObjectivetblCourse.courseObjective%TYPE;  BEGIN  -- 삭제할 과정 정보 확인  SELECT courseName, courseDuration, courseObjective  INTO v\_courseName, v\_courseDuration, v\_courseObjective  FROM tblCourse  WHERE courseId = p\_courseId;  -- 과정 삭제  DELETE FROM tblCourse WHERE courseId = p\_courseId;  -- 삭제 성공 메시지 출력  DBMS\_OUTPUT.PUT\_LINE('');  DBMS\_OUTPUT.PUT\_LINE('삭제 성공!');  DBMS\_OUTPUT.PUT\_LINE('');  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('');  DBMS\_OUTPUT.PUT\_LINE('삭제된 과정 코드 : ' || p\_courseId);  DBMS\_OUTPUT.PUT\_LINE('삭제된 과정명 : ' || v\_courseName);  DBMS\_OUTPUT.PUT\_LINE('삭제된 과정 기간 : ' || v\_courseDuration || '개월');  DBMS\_OUTPUT.PUT\_LINE('삭제된 과정 목표 : ' || v\_courseObjective);  DBMS\_OUTPUT.PUT\_LINE('');  DBMS\_OUTPUT.PUT\_LINE('======================================================');  -- 커밋 수행  COMMIT;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  -- 과정이 존재하지 않는 경우  DBMS\_OUTPUT.PUT\_LINE('삭제 실패, 과정 ID를 확인해주세요!');  WHEN OTHERS THEN  -- 기타 예외 처리  DBMS\_OUTPUT.PUT\_LINE('삭제 중 오류 발생, 다시 시도해주세요!');  ROLLBACK;  END procDeleteCourse;  /  --3. 강의 과정 수정  CREATE OR REPLACE PROCEDURE procUpdateCourse(  p\_courseId IN tblCourse.courseId%TYPE,  p\_courseName IN tblCourse.courseName%TYPE,  p\_courseDuration IN tblCourse.courseDuration%TYPE,  p\_courseObjective IN tblCourse.courseObjective%TYPE  ) AS  v\_oldNametblCourse.courseName%TYPE;  v\_oldDurationtblCourse.courseDuration%TYPE;  v\_oldObjectivetblCourse.courseObjective%TYPE;  BEGIN  -- 기존 과정 정보 확인  SELECT courseName, courseDuration, courseObjective  INTO v\_oldName, v\_oldDuration, v\_oldObjective  FROM tblCourse  WHERE courseId = p\_courseId;  -- 과정 정보 업데이트  UPDATE tblCourse  SET courseName = p\_courseName,  courseDuration = p\_courseDuration,  courseObjective = p\_courseObjective  WHERE courseId = p\_courseId;  -- 변경된 과정 정보 출력  DBMS\_OUTPUT.PUT\_LINE('');  DBMS\_OUTPUT.PUT\_LINE('수정 성공!');  DBMS\_OUTPUT.PUT\_LINE('');  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('');  DBMS\_OUTPUT.PUT\_LINE('수정된 과정 코드 : ' || p\_courseId);  DBMS\_OUTPUT.PUT\_LINE('변경 전 과정명 : ' || v\_oldName);  DBMS\_OUTPUT.PUT\_LINE('변경 후 과정명 : ' || p\_courseName);  DBMS\_OUTPUT.PUT\_LINE('변경 전 과정 기간 : ' || v\_oldDuration || '개월');  DBMS\_OUTPUT.PUT\_LINE('변경 후 과정 기간 : ' || p\_courseDuration || '개월');  DBMS\_OUTPUT.PUT\_LINE('변경 전 과정 목표 : ' || v\_oldObjective);  DBMS\_OUTPUT.PUT\_LINE('변경 후 과정 목표 : ' || p\_courseObjective);  DBMS\_OUTPUT.PUT\_LINE('');  DBMS\_OUTPUT.PUT\_LINE('======================================================');  -- 커밋 수행  COMMIT;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  -- 과정이 존재하지 않는 경우  DBMS\_OUTPUT.PUT\_LINE('수정 실패, 과정 ID를 확인해주세요!');  WHEN OTHERS THEN  -- 기타 예외 처리  DBMS\_OUTPUT.PUT\_LINE('수정 중 오류 발생, 다시 시도해주세요!');  ROLLBACK;  END procUpdateCourse;  /  --4. 강의 과정 조회  CREATE OR REPLACE PROCEDURE procGetCourses AS  -- 커서를 사용하여 여러 행을 출력할 수 있도록 설정  CURSOR cur\_course IS  SELECT courseName, courseDuration, courseObjective  FROM tblCourse;  -- 변수 선언  v\_courseNametblCourse.courseName%TYPE;  v\_courseDurationtblCourse.courseDuration%TYPE;  v\_courseObjectivetblCourse.courseObjective%TYPE;  v\_counterNUMBER := 1; -- 번호 변수 (초기값 1)  BEGIN  -- 결과 출력 시작  DBMS\_OUTPUT.PUT\_LINE('===============================================');  DBMS\_OUTPUT.PUT\_LINE(' 과정 목록 ');  DBMS\_OUTPUT.PUT\_LINE('===============================================');  DBMS\_OUTPUT.PUT\_LINE('번호 | 과정명 | 기간 | 목표 ');  DBMS\_OUTPUT.PUT\_LINE('-----------------------------------------------');  -- 커서를 열어서 데이터를 한 행씩 가져옴  FOR course\_rec IN cur\_course LOOP  DBMS\_OUTPUT.PUT\_LINE(RPAD(v\_counter, 4) || '| ' || RPAD(course\_rec.courseName, 20) || '| ' || course\_rec.courseDuration || '개월 | ' || course\_rec.courseObjective);  v\_counter := v\_counter + 1; -- 번호 증가  END LOOP;  DBMS\_OUTPUT.PUT\_LINE('===============================================');  END procGetCourses;  /  -- 강의 과정 등록 검증  EXEC procInsertCourse('스마트웹 개발을 위한 자바웹개발자 양성과정', '6', '모바일 프로그램 관련 업계 취업');  -- 강의 과정 수정 검증  EXEC procUpdateCourse(13, '프론트엔드 개발 및 자바 웹개발자 양성과정', '12', '프론트엔드 개발 관련 업계 취업');  -- 강의 과정 삭제 검증  EXEC procDeleteCourse(13);  -- 강의 과정 조회 검증  BEGIN  procGetCourses;  END;  /  /\* 과목 관리 \*/  -- 1. 과목 등록  CREATE OR REPLACE PROCEDURE procInsertSubject(  p\_subjectName IN tblSubject.subjectName%TYPE,  p\_subjectCommonStatus IN tblSubject.subjectCommonStatus%TYPE  ) AS  v\_newSubjectIdNUMBER;  BEGIN  -- 새로운 과목 ID 생성 (현재 최대 subjectId + 1)  SELECT NVL(MAX(subjectId), 0) + 1 INTO v\_newSubjectId FROM tblSubject;  -- 과목 정보 삽입  INSERT INTO tblSubject(subjectId, subjectName, subjectCommonStatus)  VALUES (v\_newSubjectId, p\_subjectName, p\_subjectCommonStatus);  -- 등록 성공 메시지 출력  DBMS\_OUTPUT.PUT\_LINE('등록 성공!');  DBMS\_OUTPUT.PUT\_LINE('');  DBMS\_OUTPUT.PUT\_LINE('=========================================');  DBMS\_OUTPUT.PUT\_LINE('');  DBMS\_OUTPUT.PUT\_LINE('과목 코드 : ' || v\_newSubjectId);  DBMS\_OUTPUT.PUT\_LINE('과목명 : ' || p\_subjectName);  DBMS\_OUTPUT.PUT\_LINE('공통 여부 : ' || p\_subjectCommonStatus);  DBMS\_OUTPUT.PUT\_LINE('');  DBMS\_OUTPUT.PUT\_LINE('=========================================');  -- 커밋 수행  COMMIT;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('등록 실패, 정보를 다시 확인해주세요!');  ROLLBACK;  END procInsertSubject;  /  -- 2. 과목 삭제  CREATE OR REPLACE PROCEDURE procDeleteSubject(  p\_subjectId IN tblSubject.subjectId%TYPE  ) AS  v\_subjectNametblSubject.subjectName%TYPE;  v\_subjectCommonStatustblSubject.subjectCommonStatus%TYPE;  BEGIN  -- 삭제할 과목 정보 확인  SELECT subjectName, subjectCommonStatus  INTO v\_subjectName, v\_subjectCommonStatus  FROM tblSubject  WHERE subjectId = p\_subjectId;  -- 과목 삭제  DELETE FROM tblSubject WHERE subjectId = p\_subjectId;  -- 삭제 성공 메시지 출력  DBMS\_OUTPUT.PUT\_LINE('삭제 성공!');  DBMS\_OUTPUT.PUT\_LINE('=========================================');  DBMS\_OUTPUT.PUT\_LINE('삭제된 과목 코드 : ' || p\_subjectId);  DBMS\_OUTPUT.PUT\_LINE('삭제된 과목명 : ' || v\_subjectName);  DBMS\_OUTPUT.PUT\_LINE('삭제된 공통 여부 : ' || v\_subjectCommonStatus);  DBMS\_OUTPUT.PUT\_LINE('=========================================');  -- 커밋 수행  COMMIT;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('삭제 실패, 과목 ID를 확인해주세요!');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('삭제 중 오류 발생, 다시 시도해주세요!');  ROLLBACK;  END procDeleteSubject;  /  -- 3. 과목 수정  CREATE OR REPLACE PROCEDURE procUpdateSubject(  p\_subjectId IN tblSubject.subjectId%TYPE,  p\_subjectName IN tblSubject.subjectName%TYPE,  p\_subjectCommonStatus IN tblSubject.subjectCommonStatus%TYPE  ) AS  v\_oldNametblSubject.subjectName%TYPE;  v\_oldCommonStatustblSubject.subjectCommonStatus%TYPE;  BEGIN  -- 기존 과목 정보 확인  SELECT subjectName, subjectCommonStatus  INTO v\_oldName, v\_oldCommonStatus  FROM tblSubject  WHERE subjectId = p\_subjectId;  -- 과목 정보 업데이트  UPDATE tblSubject  SET subjectName = p\_subjectName,  subjectCommonStatus = p\_subjectCommonStatus  WHERE subjectId = p\_subjectId;  -- 변경된 과목 정보 출력  DBMS\_OUTPUT.PUT\_LINE('수정 성공!');  DBMS\_OUTPUT.PUT\_LINE('=========================================');  DBMS\_OUTPUT.PUT\_LINE('수정된 과목 코드 : ' || p\_subjectId);  DBMS\_OUTPUT.PUT\_LINE('변경 전 과목명 : ' || v\_oldName);  DBMS\_OUTPUT.PUT\_LINE('변경 후 과목명 : ' || p\_subjectName);  DBMS\_OUTPUT.PUT\_LINE('변경 전 공통 여부 : ' || v\_oldCommonStatus);  DBMS\_OUTPUT.PUT\_LINE('변경 후 공통 여부 : ' || p\_subjectCommonStatus);  DBMS\_OUTPUT.PUT\_LINE('=========================================');  -- 커밋 수행  COMMIT;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('수정 실패, 과목 ID를 확인해주세요!');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('수정 중 오류 발생, 다시 시도해주세요!');  ROLLBACK;  END procUpdateSubject;  /  -- 4. 과목 조회  CREATE OR REPLACE PROCEDURE procGetSubjects AS  -- 커서를 사용하여 여러 행을 출력  CURSOR cur\_subject IS  SELECT subjectId, subjectName, subjectCommonStatus  FROM tblSubject;  -- 변수 선언  v\_subjectIdtblSubject.subjectId%TYPE;  v\_subjectNametblSubject.subjectName%TYPE;  v\_subjectCommonStatustblSubject.subjectCommonStatus%TYPE;  v\_counterNUMBER := 1; -- 번호 변수 (초기값 1)  BEGIN  -- 결과 출력 시작  DBMS\_OUTPUT.PUT\_LINE('=======================================');  DBMS\_OUTPUT.PUT\_LINE(' 과목 목록 ');  DBMS\_OUTPUT.PUT\_LINE('=======================================');  DBMS\_OUTPUT.PUT\_LINE('번호 | 과목 코드 | 과목명 | 공통 여부 ');  DBMS\_OUTPUT.PUT\_LINE('---------------------------------------');  -- 커서를 열어서 데이터를 한 행씩 가져옴  FOR subject\_rec IN cur\_subject LOOP  DBMS\_OUTPUT.PUT\_LINE(  RPAD(v\_counter, 4) || '| ' ||  RPAD(subject\_rec.subjectId, 10) || '| ' ||  RPAD(subject\_rec.subjectName, 15) || '| ' ||  subject\_rec.subjectCommonStatus  );  v\_counter := v\_counter + 1; -- 번호 증가  END LOOP;  DBMS\_OUTPUT.PUT\_LINE('=======================================');  END procGetSubjects;  /  -- 과목 등록 검증  EXEC procInsertSubject('React 개발', 'X');  -- 과목 수정 검증  EXEC procUpdateSubject(31, 'VueJs 개발', 'O');  -- 과목 삭제 검증  EXEC procDeleteSubject(31);  -- 과목 조회 검증  BEGIN  procGetSubjects;  END;  /  //\* 강의실 관리 \*/  -- 1. 강의실 등록  CREATE OR REPLACE PROCEDURE procInsertClassroom(  p\_classroomMaxCapacity IN tblClassroom.classroomMaxCapacity%TYPE,  p\_classroomNumber IN tblClassroom.classroomNumber%TYPE  ) AS  v\_newClassroomIdNUMBER;  v\_newClassroomNametblClassroom.classroomName%TYPE;  BEGIN  -- 새로운 강의실 ID 생성 (현재 최대 classroomId + 1)  SELECT NVL(MAX(classroomId), 0) + 1 INTO v\_newClassroomId FROM tblClassroom;  -- 강의실명 자동 생성 (ex: "8 강의실")  v\_newClassroomName := v\_newClassroomId || ' 강의실';  -- 강의실 정보 삽입  INSERT INTO tblClassroom(classroomId, classroomName, classroomMaxCapacity, classroomNumber)  VALUES (v\_newClassroomId, v\_newClassroomName, p\_classroomMaxCapacity, p\_classroomNumber);  -- 등록 성공 메시지 출력  DBMS\_OUTPUT.PUT\_LINE('등록 성공!');  DBMS\_OUTPUT.PUT\_LINE('=========================================');  DBMS\_OUTPUT.PUT\_LINE('강의실 코드 : ' || v\_newClassroomId);  DBMS\_OUTPUT.PUT\_LINE('강의실명 : ' || v\_newClassroomName);  DBMS\_OUTPUT.PUT\_LINE('수용 인원 : ' || p\_classroomMaxCapacity);  DBMS\_OUTPUT.PUT\_LINE('강의실 위치 : ' || p\_classroomNumber);  DBMS\_OUTPUT.PUT\_LINE('=========================================');  -- 커밋 수행  COMMIT;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('등록 실패, 정보를 다시 확인해주세요!');  ROLLBACK;  END procInsertClassroom;  /  -- 2. 강의실 삭제  CREATE OR REPLACE PROCEDURE procDeleteClassroom(  p\_classroomId IN tblClassroom.classroomId%TYPE  ) AS  v\_classroomNametblClassroom.classroomName%TYPE;  v\_classroomMaxCapacitytblClassroom.classroomMaxCapacity%TYPE;  v\_classroomNumbertblClassroom.classroomNumber%TYPE;  BEGIN  -- 삭제할 강의실 정보 확인  SELECT classroomName, classroomMaxCapacity, classroomNumber  INTO v\_classroomName, v\_classroomMaxCapacity, v\_classroomNumber  FROM tblClassroom  WHERE classroomId = p\_classroomId;  -- 강의실 삭제  DELETE FROM tblClassroom WHERE classroomId = p\_classroomId;  -- 삭제 성공 메시지 출력  DBMS\_OUTPUT.PUT\_LINE('삭제 성공!');  DBMS\_OUTPUT.PUT\_LINE('=========================================');  DBMS\_OUTPUT.PUT\_LINE('삭제된 강의실 코드 : ' || p\_classroomId);  DBMS\_OUTPUT.PUT\_LINE('삭제된 강의실명 : ' || v\_classroomName);  DBMS\_OUTPUT.PUT\_LINE('삭제된 수용 인원 : ' || v\_classroomMaxCapacity);  DBMS\_OUTPUT.PUT\_LINE('삭제된 강의실 위치 : ' || v\_classroomNumber);  DBMS\_OUTPUT.PUT\_LINE('=========================================');  -- 커밋 수행  COMMIT;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('삭제 실패, 강의실 ID를 확인해주세요!');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('삭제 중 오류 발생, 다시 시도해주세요!');  ROLLBACK;  END procDeleteClassroom;  /  -- 3. 강의실 수정  CREATE OR REPLACE PROCEDURE procUpdateClassroom(  p\_classroomId IN tblClassroom.classroomId%TYPE,  p\_classroomName IN tblClassroom.classroomName%TYPE,  p\_classroomMaxCapacity IN tblClassroom.classroomMaxCapacity%TYPE,  p\_classroomNumber IN tblClassroom.classroomNumber%TYPE  ) AS  v\_oldNametblClassroom.classroomName%TYPE;  v\_oldMaxCapacitytblClassroom.classroomMaxCapacity%TYPE;  v\_oldNumbertblClassroom.classroomNumber%TYPE;  BEGIN  -- 기존 강의실 정보 확인  SELECT classroomName, classroomMaxCapacity, classroomNumber  INTO v\_oldName, v\_oldMaxCapacity, v\_oldNumber  FROM tblClassroom  WHERE classroomId = p\_classroomId;  -- 강의실 정보 업데이트  UPDATE tblClassroom  SET classroomName = p\_classroomName,  classroomMaxCapacity = p\_classroomMaxCapacity,  classroomNumber = p\_classroomNumber  WHERE classroomId = p\_classroomId;  -- 변경된 강의실 정보 출력  DBMS\_OUTPUT.PUT\_LINE('수정 성공!');  DBMS\_OUTPUT.PUT\_LINE('=========================================');  DBMS\_OUTPUT.PUT\_LINE('수정된 강의실 코드 : ' || p\_classroomId);  DBMS\_OUTPUT.PUT\_LINE('변경 전 강의실명 : ' || v\_oldName);  DBMS\_OUTPUT.PUT\_LINE('변경 후 강의실명 : ' || p\_classroomName);  DBMS\_OUTPUT.PUT\_LINE('변경 전 수용 인원 : ' || v\_oldMaxCapacity);  DBMS\_OUTPUT.PUT\_LINE('변경 후 수용 인원 : ' || p\_classroomMaxCapacity);  DBMS\_OUTPUT.PUT\_LINE('변경 전 강의실 위치 : ' || v\_oldNumber);  DBMS\_OUTPUT.PUT\_LINE('변경 후 강의실 위치 : ' || p\_classroomNumber);  DBMS\_OUTPUT.PUT\_LINE('=========================================');  -- 커밋 수행  COMMIT;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('수정 실패, 강의실 ID를 확인해주세요!');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('수정 중 오류 발생, 다시 시도해주세요!');  ROLLBACK;  END procUpdateClassroom;  /  -- 4. 강의실 조회  CREATE OR REPLACE PROCEDURE procGetClassrooms AS  -- 커서를 사용하여 여러 행을 출력  CURSOR cur\_classroom IS  SELECT classroomId, classroomName, classroomMaxCapacity, classroomNumber  FROM tblClassroom;  -- 변수 선언  v\_classroomIdtblClassroom.classroomId%TYPE;  v\_classroomNametblClassroom.classroomName%TYPE;  v\_classroomMaxCapacitytblClassroom.classroomMaxCapacity%TYPE;  v\_classroomNumbertblClassroom.classroomNumber%TYPE;  v\_counterNUMBER := 1; -- 번호 변수 (초기값 1)  BEGIN  -- 결과 출력 시작  DBMS\_OUTPUT.PUT\_LINE('================================================');  DBMS\_OUTPUT.PUT\_LINE(' 강의실 목록 ');  DBMS\_OUTPUT.PUT\_LINE('================================================');  DBMS\_OUTPUT.PUT\_LINE('번호 | 강의실 코드 | 강의실명 | 수용 인원 | 강의실 위치 ');  DBMS\_OUTPUT.PUT\_LINE('------------------------------------------------');  -- 커서를 열어서 데이터를 한 행씩 가져옴  FOR classroom\_rec IN cur\_classroom LOOP  DBMS\_OUTPUT.PUT\_LINE(  RPAD(v\_counter, 4) || '| ' ||  RPAD(classroom\_rec.classroomId, 10) || '| ' ||  RPAD(classroom\_rec.classroomName, 15) || '| ' ||  RPAD(classroom\_rec.classroomMaxCapacity, 10) || '| ' ||  classroom\_rec.classroomNumber  );  v\_counter := v\_counter + 1; -- 번호 증가  END LOOP;  DBMS\_OUTPUT.PUT\_LINE('================================================');  END procGetClassrooms;  /  -- 강의실 등록 검증  EXEC procInsertClassroom(26, '807호');  -- 강의실 수정 검증  EXEC procUpdateClassroom(7, '0 강의실', 30, '800호');  -- 강의실 삭제 검증  EXEC procDeleteClassroom(7);  -- 강의실 조회 테스트  BEGIN  procGetClassrooms;  END;  /  /\* 교재 관리 \*/  -- 1. 교재 등록  CREATE OR REPLACE PROCEDURE procInsertTextbook(  p\_subjectId IN tblTextbook.subjectId%TYPE,  p\_textbookName IN tblTextbook.textbookName%TYPE,  p\_textbookPrice IN tblTextbook.textbookPrice%TYPE,  p\_textbookAuthor IN tblTextbook.textbookAuthor%TYPE,  p\_textbookPublisher IN tblTextbook.textbookPublisher%TYPE,  p\_textbookPublicationDate IN tblTextbook.textbookPublicationDate%TYPE,  p\_textbookClassification IN tblTextbook.textbookClassification%TYPE  ) AS  v\_newTextbookIdNUMBER;  BEGIN  -- 새로운 교재 ID 생성 (현재 최대 textbookId + 1)  SELECT NVL(MAX(textbookId), 0) + 1 INTO v\_newTextbookId FROM tblTextbook;  -- 교재 정보 삽입  INSERT INTO tblTextbook (textbookId, subjectId, textbookName, textbookPrice, textbookAuthor, textbookPublisher, textbookPublicationDate, textbookClassification)  VALUES (v\_newTextbookId, p\_subjectId, p\_textbookName, p\_textbookPrice, p\_textbookAuthor, p\_textbookPublisher, p\_textbookPublicationDate, p\_textbookClassification);  -- 등록 성공 메시지 출력  DBMS\_OUTPUT.PUT\_LINE('등록 성공!');  DBMS\_OUTPUT.PUT\_LINE('=========================================');  DBMS\_OUTPUT.PUT\_LINE('교재 코드 : ' || v\_newTextbookId);  DBMS\_OUTPUT.PUT\_LINE('과목 ID : ' || p\_subjectId);  DBMS\_OUTPUT.PUT\_LINE('교재명 : ' || p\_textbookName);  DBMS\_OUTPUT.PUT\_LINE('가격 : ' || p\_textbookPrice);  DBMS\_OUTPUT.PUT\_LINE('저자 : ' || p\_textbookAuthor);  DBMS\_OUTPUT.PUT\_LINE('출판사 : ' || p\_textbookPublisher);  DBMS\_OUTPUT.PUT\_LINE('발행일 : ' || p\_textbookPublicationDate);  DBMS\_OUTPUT.PUT\_LINE('분류 : ' || p\_textbookClassification);  DBMS\_OUTPUT.PUT\_LINE('=========================================');  -- 커밋 수행  COMMIT;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('등록 실패, 정보를 다시 확인해주세요!');  ROLLBACK;  END procInsertTextbook;  /  -- 2. 교재 삭제  CREATE OR REPLACE PROCEDURE procDeleteTextbook(  p\_textbookId IN tblTextbook.textbookId%TYPE  ) AS  v\_textbookNametblTextbook.textbookName%TYPE;  v\_textbookPricetblTextbook.textbookPrice%TYPE;  BEGIN  -- 삭제할 교재 정보 확인  SELECT textbookName, textbookPrice  INTO v\_textbookName, v\_textbookPrice  FROM tblTextbook  WHERE textbookId = p\_textbookId;  -- 교재 삭제  DELETE FROM tblTextbook WHERE textbookId = p\_textbookId;  -- 삭제 성공 메시지 출력  DBMS\_OUTPUT.PUT\_LINE('삭제 성공!');  DBMS\_OUTPUT.PUT\_LINE('=========================================');  DBMS\_OUTPUT.PUT\_LINE('삭제된 교재 코드 : ' || p\_textbookId);  DBMS\_OUTPUT.PUT\_LINE('삭제된 교재명 : ' || v\_textbookName);  DBMS\_OUTPUT.PUT\_LINE('삭제된 가격 : ' || v\_textbookPrice);  DBMS\_OUTPUT.PUT\_LINE('=========================================');  -- 커밋 수행  COMMIT;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('삭제 실패, 교재 ID를 확인해주세요!');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('삭제 중 오류 발생, 다시 시도해주세요!');  ROLLBACK;  END procDeleteTextbook;  /  -- 3. 교재 수정  CREATE OR REPLACE PROCEDURE procUpdateTextbook(  p\_textbookId IN tblTextbook.textbookId%TYPE,  p\_subjectId IN tblTextbook.subjectId%TYPE,  p\_textbookName IN tblTextbook.textbookName%TYPE,  p\_textbookPrice IN tblTextbook.textbookPrice%TYPE,  p\_textbookAuthor IN tblTextbook.textbookAuthor%TYPE,  p\_textbookPublisher IN tblTextbook.textbookPublisher%TYPE,  p\_textbookPublicationDate IN tblTextbook.textbookPublicationDate%TYPE,  p\_textbookClassification IN tblTextbook.textbookClassification%TYPE  ) AS  v\_oldNametblTextbook.textbookName%TYPE;  v\_oldPricetblTextbook.textbookPrice%TYPE;  BEGIN  -- 기존 교재 정보 확인  SELECT textbookName, textbookPrice  INTO v\_oldName, v\_oldPrice  FROM tblTextbook  WHERE textbookId = p\_textbookId;  -- 교재 정보 업데이트  UPDATE tblTextbook  SET subjectId = p\_subjectId,  textbookName = p\_textbookName,  textbookPrice = p\_textbookPrice,  textbookAuthor = p\_textbookAuthor,  textbookPublisher = p\_textbookPublisher,  textbookPublicationDate = p\_textbookPublicationDate,  textbookClassification = p\_textbookClassification  WHERE textbookId = p\_textbookId;  -- 변경된 교재 정보 출력  DBMS\_OUTPUT.PUT\_LINE('수정 성공!');  DBMS\_OUTPUT.PUT\_LINE('=========================================');  DBMS\_OUTPUT.PUT\_LINE('수정된 교재 코드 : ' || p\_textbookId);  DBMS\_OUTPUT.PUT\_LINE('변경 전 교재명 : ' || v\_oldName);  DBMS\_OUTPUT.PUT\_LINE('변경 후 교재명 : ' || p\_textbookName);  DBMS\_OUTPUT.PUT\_LINE('변경 전 가격 : ' || v\_oldPrice);  DBMS\_OUTPUT.PUT\_LINE('변경 후 가격 : ' || p\_textbookPrice);  DBMS\_OUTPUT.PUT\_LINE('=========================================');  -- 커밋 수행  COMMIT;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('수정 실패, 교재 ID를 확인해주세요!');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('수정 중 오류 발생, 다시 시도해주세요!');  ROLLBACK;  END procUpdateTextbook;  /  -- 4. 교재 조회  CREATE OR REPLACE PROCEDURE procGetTextbooks AS  -- 커서를 사용하여 여러 행을 출력  CURSOR cur\_textbook IS  SELECT textbookId, textbookName, textbookPrice, textbookAuthor, textbookPublisher, textbookPublicationDate, textbookClassification  FROM tblTextbook;  -- 변수 선언  v\_textbookIdtblTextbook.textbookId%TYPE;  v\_textbookNametblTextbook.textbookName%TYPE;  v\_textbookPricetblTextbook.textbookPrice%TYPE;  v\_textbookAuthortblTextbook.textbookAuthor%TYPE;  v\_textbookPublishertblTextbook.textbookPublisher%TYPE;  v\_textbookPublicationDatetblTextbook.textbookPublicationDate%TYPE;  v\_textbookClassificationtblTextbook.textbookClassification%TYPE;  v\_counterNUMBER := 1; -- 번호 변수 (초기값 1)  BEGIN  -- 결과 출력 시작  DBMS\_OUTPUT.PUT\_LINE('==============================================================');  DBMS\_OUTPUT.PUT\_LINE(' 교재 목록 ');  DBMS\_OUTPUT.PUT\_LINE('==============================================================');  DBMS\_OUTPUT.PUT\_LINE('번호 | 교재 코드 | 교재명 | 가격 | 저자 | 출판사 | 발행일 | 분류');  DBMS\_OUTPUT.PUT\_LINE('--------------------------------------------------------------');  -- 커서를 열어서 데이터를 한 행씩 가져옴  FOR textbook\_rec IN cur\_textbook LOOP  DBMS\_OUTPUT.PUT\_LINE(  RPAD(v\_counter, 4) || '| ' ||  RPAD(textbook\_rec.textbookId, 10) || '| ' ||  RPAD(textbook\_rec.textbookName, 15) || '| ' ||  RPAD(textbook\_rec.textbookPrice, 8) || '| ' ||  RPAD(textbook\_rec.textbookAuthor, 8) || '| ' ||  RPAD(textbook\_rec.textbookPublisher, 10) || '| ' ||  TO\_CHAR(textbook\_rec.textbookPublicationDate, 'YYYY-MM-DD') || '| ' ||  textbook\_rec.textbookClassification  );  v\_counter := v\_counter + 1;  END LOOP;  DBMS\_OUTPUT.PUT\_LINE('==============================================================');  END procGetTextbooks;  /  -- 1. 교재 등록 검증  EXEC procInsertTextbook(1, '실전 코드로 배우는 Vue.js', 29700, '이지온', '한빛미디어', '2024-10-10', 'it');  -- 2. 교재 수정 검증  EXEC procUpdateTextbook(31, 1, 'GPT 활용 가이드', 10000, '홍길동', 'OpenAI', '2024-10-10', 'ai');  -- 3. 교재 삭제 검증  EXEC procDeleteTextbook(31);  -- 4. 교재 조회 검증  BEGIN  procGetTextbooks;  END;  / | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 요구사항번호 | A-3.교사 계정 관리 | 작성일 | 2025-02-07 |
| 요구사항명 | 교사 계정 관리 | 작성자 | 김원중 |
| DML | | | |
| /\* 교사 계정 관리 \*/  --교사 정보 등록  CREATE OR REPLACE PROCEDURE procInsertTeacher(  p\_teacherName IN tblTeacher.teacherName%TYPE,  p\_teacherEmail IN tblTeacher.teacherEmail%TYPE,  p\_teacherPhoneNumber IN tblTeacher.teacherPhoneNumber%TYPE,  p\_teacherResidentNumberBack IN tblTeacher.teacherResidentNumberBack%TYPE  ) AS  v\_newTeacherIdNUMBER;  BEGIN  -- 새로운 teacherId 생성  SELECT NVL(MAX(teacherId), 0) + 1 INTO v\_newTeacherId FROM tblTeacher;  -- 교사 정보 삽입  INSERT INTO tblTeacher(teacherId, teacherName, teacherEmail, teacherPhoneNumber, teacherResidentNumberBack)  VALUES (v\_newTeacherId, p\_teacherName, p\_teacherEmail, p\_teacherPhoneNumber, p\_teacherResidentNumberBack);  -- 등록된 교사 정보 출력  DBMS\_OUTPUT.PUT\_LINE('교사 등록 완료');  DBMS\_OUTPUT.PUT\_LINE('-----------------------------------');  DBMS\_OUTPUT.PUT\_LINE('교사번호: ' || v\_newTeacherId);  DBMS\_OUTPUT.PUT\_LINE('이름: ' || p\_teacherName);  DBMS\_OUTPUT.PUT\_LINE('주민번호 뒷자리: ' || p\_teacherResidentNumberBack);  DBMS\_OUTPUT.PUT\_LINE('전화번호: ' || p\_teacherPhoneNumber);  DBMS\_OUTPUT.PUT\_LINE('-----------------------------------');  -- 커밋 수행  COMMIT;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('교사 등록 실패: ' || SQLERRM);  ROLLBACK;  END procInsertTeacher;  /  --교사 정보 삭제  CREATE OR REPLACE PROCEDURE procDeleteTeacher(  p\_teacherId IN tblTeacher.teacherId%TYPE  ) AS  v\_teacherNametblTeacher.teacherName%TYPE;  v\_teacherResidentNumberBacktblTeacher.teacherResidentNumberBack%TYPE;  v\_teacherPhoneNumbertblTeacher.teacherPhoneNumber%TYPE;  BEGIN  -- 삭제할 교사 정보 조회  SELECT teacherName, teacherResidentNumberBack, teacherPhoneNumber  INTO v\_teacherName, v\_teacherResidentNumberBack, v\_teacherPhoneNumber  FROM tblTeacher WHERE teacherId = p\_teacherId;  -- 교사 삭제  DELETE FROM tblTeacher WHERE teacherId = p\_teacherId;  -- 삭제된 교사 정보 출력  DBMS\_OUTPUT.PUT\_LINE('교사 삭제 완료');  DBMS\_OUTPUT.PUT\_LINE('-----------------------------------');  DBMS\_OUTPUT.PUT\_LINE('삭제된 교사번호: ' || p\_teacherId);  DBMS\_OUTPUT.PUT\_LINE('이름: ' || v\_teacherName);  DBMS\_OUTPUT.PUT\_LINE('주민번호 뒷자리: ' || v\_teacherResidentNumberBack);  DBMS\_OUTPUT.PUT\_LINE('전화번호: ' || v\_teacherPhoneNumber);  DBMS\_OUTPUT.PUT\_LINE('-----------------------------------');  -- 커밋 수행  COMMIT;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('삭제 실패: 해당 교사 ID가 존재하지 않습니다.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('교사 삭제 중 오류 발생: ' || SQLERRM);  ROLLBACK;  END procDeleteTeacher;  /  --교사 정보 수정  CREATE OR REPLACE PROCEDURE procUpdateTeacher(  p\_teacherId IN tblTeacher.teacherId%TYPE,  p\_teacherName IN tblTeacher.teacherName%TYPE,  p\_teacherEmail IN tblTeacher.teacherEmail%TYPE,  p\_teacherPhoneNumber IN tblTeacher.teacherPhoneNumber%TYPE,  p\_teacherResidentNumberBack IN tblTeacher.teacherResidentNumberBack%TYPE  ) AS  v\_oldNametblTeacher.teacherName%TYPE;  v\_oldResidentNumberBacktblTeacher.teacherResidentNumberBack%TYPE;  v\_oldPhoneNumbertblTeacher.teacherPhoneNumber%TYPE;  BEGIN  -- 기존 교사 정보 확인  SELECT teacherName, teacherResidentNumberBack, teacherPhoneNumber  INTO v\_oldName, v\_oldResidentNumberBack, v\_oldPhoneNumber  FROM tblTeacher WHERE teacherId = p\_teacherId;  -- 교사 정보 업데이트  UPDATE tblTeacher  SET teacherName = p\_teacherName,  teacherEmail = p\_teacherEmail,  teacherPhoneNumber = p\_teacherPhoneNumber,  teacherResidentNumberBack = p\_teacherResidentNumberBack  WHERE teacherId = p\_teacherId;  -- 수정된 교사 정보 출력  DBMS\_OUTPUT.PUT\_LINE('교사 정보 수정 완료');  DBMS\_OUTPUT.PUT\_LINE('-----------------------------------');  DBMS\_OUTPUT.PUT\_LINE('교사번호: ' || p\_teacherId);  DBMS\_OUTPUT.PUT\_LINE('이전 이름: ' || v\_oldName || ' -> 변경: ' || p\_teacherName);  DBMS\_OUTPUT.PUT\_LINE('이전 주민번호 뒷자리: ' || v\_oldResidentNumberBack || ' -> 변경: ' || p\_teacherResidentNumberBack);  DBMS\_OUTPUT.PUT\_LINE('이전 전화번호: ' || v\_oldPhoneNumber || ' -> 변경: ' || p\_teacherPhoneNumber);  DBMS\_OUTPUT.PUT\_LINE('-----------------------------------');  -- 커밋 수행  COMMIT;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('수정 실패: 해당 교사 ID가 존재하지 않습니다.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('교사 수정 중 오류 발생: ' || SQLERRM);  ROLLBACK;  END procUpdateTeacher;  /  --교사 정보 조회  CREATE OR REPLACE PROCEDURE procGetTeachers AS  CURSOR cur\_teacher IS  SELECT teacherId, teacherName, teacherResidentNumberBack, teacherPhoneNumber FROM tblTeacher;  v\_teacherIdtblTeacher.teacherId%TYPE;  v\_teacherNametblTeacher.teacherName%TYPE;  v\_teacherResidentNumberBacktblTeacher.teacherResidentNumberBack%TYPE;  v\_teacherPhoneNumbertblTeacher.teacherPhoneNumber%TYPE;  v\_counterNUMBER := 1;  BEGIN  DBMS\_OUTPUT.PUT\_LINE('=======================================');  DBMS\_OUTPUT.PUT\_LINE(' 교사 목록 ');  DBMS\_OUTPUT.PUT\_LINE('=======================================');  DBMS\_OUTPUT.PUT\_LINE('번호 | 교사번호 | 이름 | 주민번호뒷자리 | 전화번호 ');  DBMS\_OUTPUT.PUT\_LINE('---------------------------------------');  FOR teacher\_rec IN cur\_teacher LOOP  DBMS\_OUTPUT.PUT\_LINE(rpad(v\_counter, 3) || ' | ' || rpad(teacher\_rec.teacherId, 3) || ' | ' || teacher\_rec.teacherName || ' | ' || teacher\_rec.teacherResidentNumberBack || ' | ' || teacher\_rec.teacherPhoneNumber);  v\_counter := v\_counter + 1;  END LOOP;  DBMS\_OUTPUT.PUT\_LINE('=======================================');  END procGetTeachers;  / | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 요구사항번호 | A-4.개설 과정 관리 | 작성일 | 2025-02-07 |
| 요구사항명 | 개설 과정 관리 | 작성자 | 김원중 |
| DML | | | |
| /\* 개설 과정 관리 \*/  -- 개설 과정 등록  CREATE OR REPLACE PROCEDURE procInsertOpenedCourse(  p\_openedStartDate IN tblOpenedCourse.openedStartDate%TYPE,  p\_openedCompletionDate IN tblOpenedCourse.openedCompletionDate%TYPE,  p\_courseId IN tblOpenedCourse.courseId%TYPE,  p\_teacherId IN tblOpenedCourse.teacherId%TYPE,  p\_classroomId IN tblOpenedCourse.classroomId%TYPE,  p\_adminManagerId IN tblOpenedCourse.adminManagerId%TYPE  ) AS  v\_newOpenedCourseIdtblOpenedCourse.openedCourseId%TYPE;  BEGIN  -- 새로운 개설 과정 ID 생성  SELECT NVL(MAX(openedCourseId), 0) + 1 INTO v\_newOpenedCourseId FROM tblOpenedCourse;  -- 개설 과정 삽입  INSERT INTO tblOpenedCourse(openedCourseId, openedStartDate, openedCompletionDate, courseId, teacherId, classroomId, adminManagerId)  VALUES (v\_newOpenedCourseId, p\_openedStartDate, p\_openedCompletionDate, p\_courseId, p\_teacherId, p\_classroomId, p\_adminManagerId);  -- 등록된 개설 과정 정보 출력  DBMS\_OUTPUT.PUT\_LINE('개설 과정 등록 완료');  DBMS\_OUTPUT.PUT\_LINE('-----------------------------------');  DBMS\_OUTPUT.PUT\_LINE('개설 과정 번호: ' || v\_newOpenedCourseId);  DBMS\_OUTPUT.PUT\_LINE('개강일: ' || TO\_CHAR(p\_openedStartDate, 'YYYY-MM-DD'));  DBMS\_OUTPUT.PUT\_LINE('수료일: ' || TO\_CHAR(p\_openedCompletionDate, 'YYYY-MM-DD'));  DBMS\_OUTPUT.PUT\_LINE('-----------------------------------');  -- 커밋 수행  COMMIT;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('개설 과정 등록 실패: ' || SQLERRM);  ROLLBACK;  END procInsertOpenedCourse;  /  -- 개설 과정 삭제  CREATE OR REPLACE PROCEDURE procDeleteOpenedCourse(  p\_openedCourseId IN tblOpenedCourse.openedCourseId%TYPE  ) AS  v\_openedStartDatetblOpenedCourse.openedStartDate%TYPE;  v\_openedCompletionDatetblOpenedCourse.openedCompletionDate%TYPE;  v\_courseIdtblOpenedCourse.courseId%TYPE;  v\_teacherIdtblOpenedCourse.teacherId%TYPE;  v\_classroomIdtblOpenedCourse.classroomId%TYPE;  v\_adminManagerIdtblOpenedCourse.adminManagerId%TYPE;  BEGIN  -- 삭제할 개설 과정 정보 조회  SELECT openedStartDate, openedCompletionDate, courseId, teacherId, classroomId, adminManagerId  INTO v\_openedStartDate, v\_openedCompletionDate, v\_courseId, v\_teacherId, v\_classroomId, v\_adminManagerId  FROM tblOpenedCourse WHERE openedCourseId = p\_openedCourseId;  -- 개설 과정 삭제  DELETE FROM tblOpenedCourse WHERE openedCourseId = p\_openedCourseId;  -- 삭제된 개설 과정 정보 출력  DBMS\_OUTPUT.PUT\_LINE('개설 과정 삭제 완료');  DBMS\_OUTPUT.PUT\_LINE('-----------------------------------');  DBMS\_OUTPUT.PUT\_LINE('삭제된 개설 과정 번호: ' || p\_openedCourseId);  DBMS\_OUTPUT.PUT\_LINE('개강일: ' || TO\_CHAR(v\_openedStartDate, 'YYYY-MM-DD'));  DBMS\_OUTPUT.PUT\_LINE('수료일: ' || TO\_CHAR(v\_openedCompletionDate, 'YYYY-MM-DD'));  DBMS\_OUTPUT.PUT\_LINE('-----------------------------------');  -- 커밋 수행  COMMIT;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('삭제 실패: 해당 개설 과정 ID가 존재하지 않습니다.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('개설 과정 삭제 중 오류 발생: ' || SQLERRM);  ROLLBACK;  END procDeleteOpenedCourse;  /  -- 개설 과정 수정  CREATE OR REPLACE PROCEDURE procUpdateOpenedCourse(  p\_openedCourseId IN tblOpenedCourse.openedCourseId%TYPE,  p\_openedStartDate IN tblOpenedCourse.openedStartDate%TYPE,  p\_openedCompletionDate IN tblOpenedCourse.openedCompletionDate%TYPE,  p\_courseId IN tblOpenedCourse.courseId%TYPE,  p\_teacherId IN tblOpenedCourse.teacherId%TYPE,  p\_classroomId IN tblOpenedCourse.classroomId%TYPE,  p\_adminManagerId IN tblOpenedCourse.adminManagerId%TYPE  ) AS  v\_oldOpenedStartDatetblOpenedCourse.openedStartDate%TYPE;  v\_oldOpenedCompletionDatetblOpenedCourse.openedCompletionDate%TYPE;  v\_oldCourseIdtblOpenedCourse.courseId%TYPE;  v\_oldTeacherIdtblOpenedCourse.teacherId%TYPE;  v\_oldClassroomIdtblOpenedCourse.classroomId%TYPE;  v\_oldAdminManagerIdtblOpenedCourse.adminManagerId%TYPE;  BEGIN  -- 기존 개설 과정 정보 확인  SELECT openedStartDate, openedCompletionDate, courseId, teacherId, classroomId, adminManagerId  INTO v\_oldOpenedStartDate, v\_oldOpenedCompletionDate, v\_oldCourseId, v\_oldTeacherId, v\_oldClassroomId, v\_oldAdminManagerId  FROM tblOpenedCourse WHERE openedCourseId = p\_openedCourseId;  -- 개설 과정 정보 업데이트  UPDATE tblOpenedCourse  SET openedStartDate = p\_openedStartDate,  openedCompletionDate = p\_openedCompletionDate,  courseId = p\_courseId,  teacherId = p\_teacherId,  classroomId = p\_classroomId,  adminManagerId = p\_adminManagerId  WHERE openedCourseId = p\_openedCourseId;  -- 수정된 개설 과정 정보 출력  DBMS\_OUTPUT.PUT\_LINE('개설 과정 수정 완료');  DBMS\_OUTPUT.PUT\_LINE('-----------------------------------');  DBMS\_OUTPUT.PUT\_LINE('개설 과정 번호: ' || p\_openedCourseId);  DBMS\_OUTPUT.PUT\_LINE('이전 개강일: ' || TO\_CHAR(v\_oldOpenedStartDate, 'YYYY-MM-DD') || ' -> 변경: ' || TO\_CHAR(p\_openedStartDate, 'YYYY-MM-DD'));  DBMS\_OUTPUT.PUT\_LINE('이전 수료일: ' || TO\_CHAR(v\_oldOpenedCompletionDate, 'YYYY-MM-DD') || ' -> 변경: ' || TO\_CHAR(p\_openedCompletionDate, 'YYYY-MM-DD'));  DBMS\_OUTPUT.PUT\_LINE('-----------------------------------');  -- 커밋 수행  COMMIT;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('수정 실패: 해당 개설 과정 ID가 존재하지 않습니다.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('개설 과정 수정 중 오류 발생: ' || SQLERRM);  ROLLBACK;  END procUpdateOpenedCourse;  /  -- 개설 과정 조회  CREATE OR REPLACE PROCEDURE procGetOpenedCourses AS  CURSOR cur\_opened\_course IS  SELECT openedCourseId, openedStartDate, openedCompletionDate, courseId, teacherId, classroomId, adminManagerId  FROM tblOpenedCourse;  v\_openedCourseIdtblOpenedCourse.openedCourseId%TYPE;  v\_openedStartDatetblOpenedCourse.openedStartDate%TYPE;  v\_openedCompletionDatetblOpenedCourse.openedCompletionDate%TYPE;  v\_courseIdtblOpenedCourse.courseId%TYPE;  v\_teacherIdtblOpenedCourse.teacherId%TYPE;  v\_classroomIdtblOpenedCourse.classroomId%TYPE;  v\_adminManagerIdtblOpenedCourse.adminManagerId%TYPE;  v\_counterNUMBER := 1;  BEGIN  DBMS\_OUTPUT.PUT\_LINE('=======================================');  DBMS\_OUTPUT.PUT\_LINE(' 개설 과정 목록 ');  DBMS\_OUTPUT.PUT\_LINE('=======================================');  DBMS\_OUTPUT.PUT\_LINE('번호 | 개설 과정 번호 | 개강일 | 수료일 | 과정 ID | 교사 ID | 강의실 ID | 관리자 ID');  DBMS\_OUTPUT.PUT\_LINE('---------------------------------------');  FOR course\_rec IN cur\_opened\_course LOOP  DBMS\_OUTPUT.PUT\_LINE(rpad(v\_counter, 3) || ' | ' || rpad(course\_rec.openedCourseId, 3) || ' | ' || TO\_CHAR(course\_rec.openedStartDate, 'YYYY-MM-DD') || ' | ' || TO\_CHAR(course\_rec.openedCompletionDate, 'YYYY-MM-DD') || ' | ' || rpad(course\_rec.courseId, 3) || ' | ' || rpad(course\_rec.teacherId, 3) || ' | ' || rpad(course\_rec.classroomId, 3) || ' | ' || rpad(course\_rec.adminManagerId, 3));  v\_counter := v\_counter + 1;  END LOOP;  DBMS\_OUTPUT.PUT\_LINE('=======================================');  END procGetOpenedCourses;  /  -- 개설 과정 등록 검증  EXEC procInsertOpenedCourse('2025-05-16', '2025-11-14', 8, 9, 4, 2);  -- 개설 과정 수정 검증  EXEC procUpdateOpenedCourse(13, '2025-05-15', '2025-11-13', 9, 7, 1, 1);  -- 개설 과정 삭제 검증  EXEC procDeleteOpenedCourse(13);  -- 개설 과정 조회 검증  EXEC procGetOpenedCourse(13); | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 요구사항번호 | A-5. 개설 과목 관리 | 작성일 | 2025-02-07 |
| 요구사항명 | 교사 계정 관리 | 작성자 | 김원중 |
| DML | | | |
| /\* 개설 과목 관리 \*/  --개설 과목 등록  CREATE OR REPLACE PROCEDURE procInsertOpenedSubject(  p\_subjectStartDate IN DATE,  p\_subjectEndDate IN DATE,  p\_subjectId IN tblSubject.subjectId%TYPE,  p\_openedCourseId IN tblOpenedCourse.openedCourseId%TYPE  ) AS  v\_subjectNametblSubject.subjectname%TYPE;  v\_textbookIdtblTextbook.textbookId%TYPE;  BEGIN  -- 과목명과 교재번호 조회  SELECT s.subjectname, tb.textbookId  INTO v\_subjectName, v\_textbookId  FROM tblSubject s  JOIN tblTextbook tb ON tb.subjectId = s.subjectId  WHERE s.subjectId = p\_subjectId;  -- 개설과목 등록  INSERT INTO tblOpenedSubject (openedSubjectId, subjectStartDate, subjectEndDate, subjectId, openedCourseId)  VALUES (  (SELECT max(openedSubjectId) + 1 FROM tblOpenedSubject),  p\_subjectStartDate, p\_subjectEndDate, p\_subjectId, p\_openedCourseId  );  COMMIT; -- 변경 사항을 저장  -- 등록된 개설 과목 정보 출력  DBMS\_OUTPUT.PUT\_LINE('===========================================');  DBMS\_OUTPUT.PUT\_LINE('개설 과목 등록 완료');  DBMS\_OUTPUT.PUT\_LINE('과목명: ' || v\_subjectName);  DBMS\_OUTPUT.PUT\_LINE('과목시작일: ' || TO\_CHAR(p\_subjectStartDate, 'yyyy-mm-dd'));  DBMS\_OUTPUT.PUT\_LINE('과목종료일: ' || TO\_CHAR(p\_subjectEndDate, 'yyyy-mm-dd'));  DBMS\_OUTPUT.PUT\_LINE('교재번호: ' || v\_textbookId);  DBMS\_OUTPUT.PUT\_LINE('과정번호: ' || p\_openedCourseId);  DBMS\_OUTPUT.PUT\_LINE('===========================================');  END procInsertOpenedSubject;  /  --개설 과목 수정  CREATE OR REPLACE PROCEDURE procUpdateOpenedSubject(  p\_openedSubjectId IN tblOpenedSubject.openedSubjectId%TYPE,  p\_subjectStartDate IN DATE,  p\_subjectEndDate IN DATE,  p\_subjectId IN tblSubject.subjectId%TYPE,  p\_openedCourseId IN tblOpenedCourse.openedCourseId%TYPE  ) AS  v\_subjectNametblSubject.subjectname%TYPE;  v\_textbookIdtblTextbook.textbookId%TYPE;  BEGIN  -- 과목명과 교재번호 조회  SELECT s.subjectname, tb.textbookId  INTO v\_subjectName, v\_textbookId  FROM tblSubject s  JOIN tblTextbook tb ON tb.subjectId = s.subjectId  WHERE s.subjectId = p\_subjectId;  -- 개설과목 수정  UPDATE tblOpenedSubject  SET  subjectStartDate = p\_subjectStartDate,  subjectEndDate = p\_subjectEndDate,  subjectId = p\_subjectId,  openedCourseId = p\_openedCourseId  WHERE openedSubjectId = p\_openedSubjectId;  COMMIT; -- 변경 사항을 저장  -- 수정된 개설 과목 정보 출력  DBMS\_OUTPUT.PUT\_LINE('===========================================');  DBMS\_OUTPUT.PUT\_LINE('개설 과목 수정 완료');  DBMS\_OUTPUT.PUT\_LINE('과목명: ' || v\_subjectName);  DBMS\_OUTPUT.PUT\_LINE('과목시작일: ' || TO\_CHAR(p\_subjectStartDate, 'yyyy-mm-dd'));  DBMS\_OUTPUT.PUT\_LINE('과목종료일: ' || TO\_CHAR(p\_subjectEndDate, 'yyyy-mm-dd'));  DBMS\_OUTPUT.PUT\_LINE('교재번호: ' || v\_textbookId);  DBMS\_OUTPUT.PUT\_LINE('과정번호: ' || p\_openedCourseId);  DBMS\_OUTPUT.PUT\_LINE('===========================================');  END procUpdateOpenedSubject;  /  --개설 과목 삭제  CREATE OR REPLACE PROCEDURE procDeleteOpenedSubject(  p\_openedSubjectId IN tblOpenedSubject.openedSubjectId%TYPE  ) AS  v\_subjectNametblSubject.subjectname%TYPE;  v\_textbookIdtblTextbook.textbookId%TYPE;  BEGIN  -- 삭제될 과목명과 교재번호 조회  SELECT s.subjectname, tb.textbookId  INTO v\_subjectName, v\_textbookId  FROM tblOpenedSubjectos  JOIN tblSubject s ON os.subjectId = s.subjectId  JOIN tblTextbook tb ON tb.subjectId = s.subjectId  WHERE os.openedSubjectId = p\_openedSubjectId;  -- 개설과목 삭제  DELETE FROM tblOpenedSubject  WHERE openedSubjectId = p\_openedSubjectId;  COMMIT; -- 변경 사항을 저장  -- 삭제된 개설 과목 정보 출력  DBMS\_OUTPUT.PUT\_LINE('===========================================');  DBMS\_OUTPUT.PUT\_LINE('개설 과목 삭제 완료');  DBMS\_OUTPUT.PUT\_LINE('과목명: ' || v\_subjectName);  DBMS\_OUTPUT.PUT\_LINE('교재번호: ' || v\_textbookId);  DBMS\_OUTPUT.PUT\_LINE('===========================================');  END procDeleteOpenedSubject;  /  --개설 과목 조회  CREATE OR REPLACE PROCEDURE procGetOpenedSubjectDetails(  p\_openedSubjectId IN tblOpenedSubject.openedSubjectId%TYPE  ) AS  v\_subjectNametblSubject.subjectname%TYPE;  v\_textbookIdtblTextbook.textbookId%TYPE;  v\_courseNametblCourse.coursename%TYPE;  v\_subjectStartDatetblOpenedSubject.subjectStartDate%TYPE;  v\_subjectEndDatetblOpenedSubject.subjectEndDate%TYPE;  BEGIN  -- 과목명, 교재번호, 과정명 조회  BEGIN  SELECT s.subjectname, tb.textbookId, c.coursename,  os.subjectStartDate, os.subjectEndDate  INTO v\_subjectName, v\_textbookId, v\_courseName, v\_subjectStartDate, v\_subjectEndDate  FROM tblOpenedSubjectos  JOIN tblSubject s ON os.subjectId = s.subjectId  JOIN tblTextbook tb ON tb.subjectId = s.subjectId  JOIN tblCourse c ON os.openedCourseId = c.courseId  WHERE os.openedSubjectId = p\_openedSubjectId;  -- 조회된 개설 과목 정보 출력  DBMS\_OUTPUT.PUT\_LINE('===========================================');  DBMS\_OUTPUT.PUT\_LINE('개설 과목 조회 결과');  DBMS\_OUTPUT.PUT\_LINE('과목명: ' || v\_subjectName);  DBMS\_OUTPUT.PUT\_LINE('과목시작일: ' || TO\_CHAR(v\_subjectStartDate, 'yyyy-mm-dd'));  DBMS\_OUTPUT.PUT\_LINE('과목종료일: ' || TO\_CHAR(v\_subjectEndDate, 'yyyy-mm-dd'));  DBMS\_OUTPUT.PUT\_LINE('교재번호: ' || v\_textbookId);  DBMS\_OUTPUT.PUT\_LINE('과정명: ' || v\_courseName);  DBMS\_OUTPUT.PUT\_LINE('===========================================');  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  -- 조회 결과가 없을 경우 메시지 출력  DBMS\_OUTPUT.PUT\_LINE('해당 과목은 등록되지 않은 과목입니다.');  END;  END procGetOpenedSubjectDetails;  /  --개설 과목 등록 검증  EXEC procInsertOpenedSubject('2025-10-10', '2025-11-10', 2, 12);  --개설 과목 수정 검증  EXEC procUpdateOpenedSubject(78, '2025-10-11', '2025-11-11', 2, 12);  --개설 과목 삭제 검증  EXEC procDeleteOpenedSubject(78);  --개설 과목 조회 검증  EXEC procGetOpenedSubjectDetails(78); | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 요구사항번호 | A-6.교육생 관리 | 작성일 | 2025-02-07 |
| 요구사항명 | 교육생 관리 | 작성자 | 김원중 |
| DML | | | |
| /\* 교육생 관리 \*/  --1. 교육생 등록  CREATE OR REPLACE PROCEDURE procInsertStudent(  p\_studentName IN tblStudent.studentName%TYPE,  p\_studentEmail IN tblStudent.studentEmail%TYPE,  p\_studentPhoneNumber IN tblStudent.studentPhoneNumber%TYPE,  p\_studentResidentNumberBack IN tblStudent.studentResidentNumberBack%TYPE,  p\_studentCourseId IN tblStudent.studentCourseId%TYPE  ) AS  BEGIN  -- 교육생 등록  INSERT INTO tblStudent (studentId, studentRegistrationDate, studentMajor, studentName, studentEmail, studentPhoneNumber, studentResidentNumberBack, studentCompletionStatus, studentCourseId)  VALUES ((SELECT MAX(studentId) + 1 FROM tblStudent), SYSDATE, 'O', p\_studentName, p\_studentEmail, p\_studentPhoneNumber, p\_studentResidentNumberBack, '수강대기', p\_studentCourseId);  DBMS\_OUTPUT.PUT\_LINE('교육생 등록 완료: ' || p\_studentName);  END procInsertStudent;  /  --2. 교육생 조회  CREATE OR REPLACE PROCEDURE procGetStudentDetails(  p\_studentId IN tblStudent.studentId%TYPE  ) AS  v\_studentNametblStudent.studentName%TYPE;  v\_studentResidentNumberBacktblStudent.studentResidentNumberBack%TYPE;  v\_studentPhoneNumbertblStudent.studentPhoneNumber%TYPE;  v\_studentRegistrationDatetblStudent.studentRegistrationDate%TYPE;  v\_courseCountNUMBER;  BEGIN  -- 교육생 정보 조회  SELECT studentName, studentResidentNumberBack, studentPhoneNumber, studentRegistrationDate  INTO v\_studentName, v\_studentResidentNumberBack, v\_studentPhoneNumber, v\_studentRegistrationDate  FROM tblStudent  WHERE studentId = p\_studentId;  -- 수강 신청 횟수 조회  SELECT COUNT(\*) INTO v\_courseCount  FROM tblStudent  WHERE studentId = p\_studentId;  -- 교육생 정보 출력  DBMS\_OUTPUT.PUT\_LINE('===========================================');  DBMS\_OUTPUT.PUT\_LINE('교육생 정보');  DBMS\_OUTPUT.PUT\_LINE('이름: ' || v\_studentName);  DBMS\_OUTPUT.PUT\_LINE('주민번호 뒷자리: ' || v\_studentResidentNumberBack);  DBMS\_OUTPUT.PUT\_LINE('전화번호: ' || v\_studentPhoneNumber);  DBMS\_OUTPUT.PUT\_LINE('등록일: ' || TO\_CHAR(v\_studentRegistrationDate, 'yyyy-mm-dd'));  DBMS\_OUTPUT.PUT\_LINE('수강 신청 횟수: ' || v\_courseCount);  DBMS\_OUTPUT.PUT\_LINE('===========================================');  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('해당 교육생 정보가 없습니다.');  END procGetStudentDetails;  /  --3. 수강 정보 상세 조회  CREATE OR REPLACE PROCEDURE procGetCourseDetails(  p\_studentId IN tblStudent.studentId%TYPE  ) AS  CURSOR c\_courseDetails IS  SELECT s.studentName, c.coursename AS 개설과정명,  TO\_CHAR(oc.openedStartDate, 'yyyymmdd') || ' ~ ' || TO\_CHAR(oc.openedCompletionDate, 'yyyymmdd') AS 과정기간,  cr.classroomname AS 강의실명,  t.teachername AS 담당교사명,  a.adminname AS 담당매니저명,  s.studentCompletionStatus AS 수강상태  FROM tblStudent s  INNER JOIN tblOpenedCourseoc ON s.studentCourseId = oc.openedCourseId  INNER JOIN tblClassroomcr ON cr.classroomId = oc.classroomId  INNER JOIN tblTeacher t ON oc.teacherId = t.teacherId  INNER JOIN tblAdmin a ON a.adminManagerId = oc.adminManagerId  INNER JOIN tblCourse c ON c.courseId = oc.courseId  WHERE s.studentId = p\_studentId;  BEGIN  FOR rec IN c\_courseDetails LOOP  DBMS\_OUTPUT.PUT\_LINE('===========================================');  DBMS\_OUTPUT.PUT\_LINE('교육생 이름: ' || rec.studentName);  DBMS\_OUTPUT.PUT\_LINE('개설 과정명: ' || rec.개설과정명);  DBMS\_OUTPUT.PUT\_LINE('과정 기간: ' || rec.과정기간);  DBMS\_OUTPUT.PUT\_LINE('강의실: ' || rec.강의실명);  DBMS\_OUTPUT.PUT\_LINE('담당 교사: ' || rec.담당교사명);  DBMS\_OUTPUT.PUT\_LINE('담당 매니저: ' || rec.담당매니저명);  DBMS\_OUTPUT.PUT\_LINE('수강 상태: ' || rec.수강상태);  DBMS\_OUTPUT.PUT\_LINE('===========================================');  END LOOP;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('해당 교육생의 수강 정보가 없습니다.');  END procGetCourseDetails;  /  -- 교육생 등록 검증  EXEC procInsertStudent('이수만', 'sumani@kakao.com', '010-5616-2124', '1240124', 12);  -- 교육생 정보 조회 검증  EXEC procGetStudentDetails(1);  -- 교육생 수강 정보 상세 조회  EXEC procGetCourseDetails(1);  --4. 개설 과정의 수강 상태(중/예정/종료) 별 교육생 정보 출력  CREATE OR REPLACE PROCEDURE procGetStudentByCourseStatus(  p\_courseStatus IN VARCHAR2 -- '예정', '진행', '종료' 중 하나 입력  ) AS  BEGIN  FOR rec IN (  SELECT s.studentName AS 교육생이름,  s.studentResidentNumberBack AS 주민번호뒷자리,  s.studentPhoneNumber AS 전화번호,  s.studentRegistrationDate AS 등록일,  c.courseName AS 과정명,  oc.openedStartDate AS 과정시작일,  oc.openedCompletionDate AS 과정종료일  FROM tblStudent s  INNER JOIN tblOpenedCourseoc ON s.studentCourseId = oc.openedCourseId  INNER JOIN tblCourse c ON c.courseId = oc.courseId  WHERE (p\_courseStatus = '예정' AND SYSDATE <oc.openedStartDate)  OR (p\_courseStatus = '진행' AND SYSDATE BETWEEN oc.openedStartDate AND oc.openedCompletionDate)  OR (p\_courseStatus = '종료' AND SYSDATE >oc.openedCompletionDate)  ) LOOP  DBMS\_OUTPUT.PUT\_LINE('===========================================');  DBMS\_OUTPUT.PUT\_LINE('교육생 이름: ' || rec.교육생이름);  DBMS\_OUTPUT.PUT\_LINE('주민번호 뒷자리: ' || rec.주민번호뒷자리);  DBMS\_OUTPUT.PUT\_LINE('전화번호: ' || rec.전화번호);  DBMS\_OUTPUT.PUT\_LINE('등록일: ' || TO\_CHAR(rec.등록일, 'YYYY-MM-DD'));  DBMS\_OUTPUT.PUT\_LINE('과정명: ' || rec.과정명);  DBMS\_OUTPUT.PUT\_LINE('과정 시작일: ' || TO\_CHAR(rec.과정시작일, 'YYYY-MM-DD'));  DBMS\_OUTPUT.PUT\_LINE('과정 종료일: ' || TO\_CHAR(rec.과정종료일, 'YYYY-MM-DD'));  DBMS\_OUTPUT.PUT\_LINE('===========================================');  END LOOP;  -- 데이터가 없을 경우 처리  IF SQL%ROWCOUNT = 0 THEN  DBMS\_OUTPUT.PUT\_LINE('해당 상태의 교육생 정보가 없습니다.');  END IF;  END procGetStudentByCourseStatus;  /  --수강 예정  EXEC procGetStudentByCourseStatus('예정');  --수강 중  EXEC procGetStudentByCourseStatus('진행');  --수강 종료  EXEC procGetStudentByCourseStatus('종료');  --5.교육생 정보 수정  CREATE OR REPLACE PROCEDURE procUpdateStudent(  p\_studentId IN tblStudent.studentId%TYPE,  p\_studentName IN tblStudent.studentName%TYPE,  p\_studentResidentNumberBack IN tblStudent.studentResidentNumberBack%TYPE,  p\_studentPhoneNumber IN tblStudent.studentPhoneNumber%TYPE,  p\_studentRegistrationDate IN DATE  ) AS  v\_countNUMBER;  v\_updatedNametblStudent.studentName%TYPE;  v\_updatedResidentNumberBacktblStudent.studentResidentNumberBack%TYPE;  v\_updatedPhoneNumbertblStudent.studentPhoneNumber%TYPE;  v\_updatedRegistrationDatetblStudent.studentRegistrationDate%TYPE;  BEGIN  -- 해당 학생이 존재하는지 확인  SELECT COUNT(\*)  INTO v\_count  FROM tblStudent  WHERE studentId = p\_studentId;  IF v\_count = 0 THEN  DBMS\_OUTPUT.PUT\_LINE('오류: 해당 studentId(' || p\_studentId || ')의 교육생이 존재하지 않습니다.');  RETURN;  END IF;  -- 교육생 정보 업데이트  UPDATE tblStudent  SET studentName = p\_studentName,  studentResidentNumberBack = p\_studentResidentNumberBack,  studentPhoneNumber = p\_studentPhoneNumber,  studentRegistrationDate = p\_studentRegistrationDate  WHERE studentId = p\_studentId;  SELECT studentName, studentResidentNumberBack, studentPhoneNumber, studentRegistrationDate  INTO v\_updatedName, v\_updatedResidentNumberBack, v\_updatedPhoneNumber, v\_updatedRegistrationDate  FROM tblStudent  WHERE studentId = p\_studentId;  DBMS\_OUTPUT.PUT\_LINE('===========================================');  DBMS\_OUTPUT.PUT\_LINE('교육생 정보 수정 완료!');  DBMS\_OUTPUT.PUT\_LINE('학생 ID: ' || p\_studentId);  DBMS\_OUTPUT.PUT\_LINE('이름: ' || v\_updatedName);  DBMS\_OUTPUT.PUT\_LINE('주민번호 뒷자리: ' || v\_updatedResidentNumberBack);  DBMS\_OUTPUT.PUT\_LINE('전화번호: ' || v\_updatedPhoneNumber);  DBMS\_OUTPUT.PUT\_LINE('등록일: ' || TO\_CHAR(v\_updatedRegistrationDate, 'YYYY-MM-DD'));  DBMS\_OUTPUT.PUT\_LINE('===========================================');  COMMIT;  END procUpdateStudent;  /  --교육생 정보 수정 검증  CALL procUpdateStudent(  1,  '홍길동',  '1111111',  '010-5180-1258',  DATE '2024-04-07'  );  --6. 교육생 수료/중도 탈락 처리  CREATE OR REPLACE PROCEDURE procUpdateStudentCompletionStatus(  p\_studentId IN tblStudent.studentId%TYPE,  p\_status IN VARCHAR2, -- '수료완료' 또는 '중도포기'  p\_giveupReason IN VARCHAR2 DEFAULT NULL -- 중도포기 사유 (수료완료일 경우 NULL)  ) AS  v\_countNUMBER;  v\_updatedStatustblStudent.studentCompletionStatus%TYPE;  BEGIN  -- 해당 학생이 존재하는지 확인  SELECT COUNT(\*)  INTO v\_count  FROM tblStudent  WHERE studentId = p\_studentId;  IF v\_count = 0 THEN  DBMS\_OUTPUT.PUT\_LINE('오류: 해당 studentId(' || p\_studentId || ')의 교육생이 존재하지 않습니다.');  RETURN;  END IF;  -- 교육생 수료 상태 업데이트  UPDATE tblStudent  SET studentCompletionStatus = p\_status  WHERE studentId = p\_studentId;  -- 중도 포기 처리일 경우 tblGiveup 테이블에 기록  IF p\_status = '중도포기' THEN  INSERT INTO tblGiveup (studentId, giveupDate, giveupReason)  VALUES (p\_studentId, SYSDATE, p\_giveupReason);  END IF;  SELECT studentCompletionStatus  INTO v\_updatedStatus  FROM tblStudent  WHERE studentId = p\_studentId;  DBMS\_OUTPUT.PUT\_LINE('===========================================');  DBMS\_OUTPUT.PUT\_LINE('교육생 ID: ' || p\_studentId);  DBMS\_OUTPUT.PUT\_LINE('수료 상태 변경 완료! 현재 상태: ' || v\_updatedStatus);  IF p\_status = '중도포기' THEN  DBMS\_OUTPUT.PUT\_LINE('중도 포기 사유: ' || p\_giveupReason);  END IF;  DBMS\_OUTPUT.PUT\_LINE('===========================================');  COMMIT;  END procUpdateStudentCompletionStatus;  /  --수료 완료일 경우 검증  EXEC procUpdateStudentCompletionStatus(1, '수료대기');  --중도 포기일 경우 검증  EXEC procUpdateStudentCompletionStatus(318, '중도포기', '학습 난이도');  SELECT \* FROM tblStudent WHERE studentId IN (1, 318);  SELECT \* FROM tblGiveup WHERE studentId = 318; | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 요구사항번호 | A-7.시험 관리 및 성적 조회 | 작성일 | 2025-02-07 |
| 요구사항명 | 시험 관리 및 성적 조회 | 작성자 | 김원중 |
| DML | | | |
| /\* 시험 관리 및 성적 조회 \*/  --1. 특정 개설 과정 선택 후 개설 과목 정보 출력  create or replace procedure procGetSubjectInfoByCourse(  popenedcourseid number  )  is  cursor vcursor  is  select os.openedsubjectid as 개설과목번호  , c.coursename as 과정명  , to\_char(oc.openedstartdate, 'yyyymmdd') as 과정기간\_start  , to\_char(oc.openedcompletiondate, 'yyyymmdd') as 과정기간\_end  , crm.classroomname as 강의실명  , s.subjectname as 과목명  , to\_char(os.subjectStartDate, 'yyyymmdd') as 과목기간\_start  , to\_char(os.subjectEndDate, 'yyyymmdd') as 과목기간\_end  , tb.textbookname as 교재명  , os.subjectattendancescoreweight as 출결배점  , os.subjectwrittenscoreweight as 필기배점  , os.subjectpracticalscoreweight as 실기배점  , case when os.openedsubjectid in (  --성적 등록이 안된 과목  select distinct openedsubjectId  from (  --개인 성적 정보  select g.gradeId  , os.openedsubjectId  , s.studentId  , s.studentCourseId  , s.studentName  , s.studentCompletionStatus  , gradeWrittenScore  , gradePracticalScore  , gradeAttendanceScore  from tblStudent s  inner join tblOpenedCourseoc  on s.studentCourseId = oc.openedCourseId  inner join tblOpenedSubjectos  on os.openedCourseId = oc.openedCourseId  left outer join tblGrade g  on g.openedsubjectId = os.openedsubjectId  and g.studentId = s.studentId  ) x  where ( x.gradeWrittenScore is null  or x.gradePracticalScore is null  or x.gradeAttendanceScore is null )  and x.studentCompletionStatus != '중도포기'  ) then 'X'  else 'O'  end as 성적등록여부  from tblOpenedCourseoc --개설과정  inner join tblCourse c --강의과정  on oc.courseid = c.courseid  inner join tblOpenedSubjectos --개설과목  on os.openedcourseid = oc.openedcourseid  inner join tblSubject s --과목  on s.subjectid = os.subjectid  inner join tblClassroomcrm --강의실  on crm.classroomid = oc.classroomid  inner join tbltextbook tb --교재  on tb.subjectId = s.subjectId  where oc.openedcourseid = popenedcourseid  order by oc.openedcourseid, os.openedsubjectid  ;  v\_openedsubjectidtblOpenedSubject.openedsubjectid%type;  v\_coursenametblCourse.coursename%type;  v\_openedstartdatetblOpenedCourse.openedstartdate%type;  v\_openedcompletiondatetblOpenedCourse.openedcompletiondate%type;  v\_classroomnametblClassroom.classroomname%type;  v\_subjectnametblSubject.subjectname%type;  v\_subjectStartDatetblOpenedSubject.subjectStartDate%type;  v\_subjectEndDatetblOpenedSubject.subjectEndDate%type;  v\_textbooknametbltextbook.textbookname%type;  v\_subjectattendancescoreweighttblOpenedSubject.subjectattendancescoreweight%type;  v\_subjectwrittenscoreweighttblOpenedSubject.subjectwrittenscoreweight%type;  v\_subjectpracticalscoreweighttblOpenedSubject.subjectpracticalscoreweight%type;  v\_openedsubjectYnvarchar(2);  begin  open vcursor;  dbms\_output.put\_line('=====================================================================================================================================' );  dbms\_output.put\_line('|| 개설과목번호 || 과정명 || 과정기간 || 강의실명 || 과목명 || 과목기간 || 교재명 || 출결배점 || 필기배점 || 실기배점 || 성적등록여부' );  dbms\_output.put\_line('=====================================================================================================================================' );  loop  fetch vcursor into v\_openedsubjectid, v\_coursename, v\_openedstartdate, v\_openedcompletiondate, v\_classroomname, v\_subjectname  , v\_subjectStartDate, v\_subjectEndDate, v\_textbookname, v\_subjectattendancescoreweight, v\_subjectwrittenscoreweight  , v\_subjectpracticalscoreweight, v\_openedsubjectYn;  exit when vcursor%notfound;  dbms\_output.put\_line(v\_openedsubjectid || ' || ' || v\_coursename || ' || '|| to\_char(v\_openedstartdate, 'yyyymmdd') || ' ~ ' || to\_char(v\_openedcompletiondate , 'yyyymmdd')  || ' || ' || v\_classroomname || ' || ' || v\_subjectname || ' || ' || to\_char(v\_subjectStartDate, 'yyyymmdd') || ' ~ ' || to\_char(v\_subjectEndDate , 'yyyymmdd')  || ' || ' || v\_textbookname || ' || ' || v\_subjectattendancescoreweight || ' || ' || v\_subjectwrittenscoreweight  || ' || ' || v\_subjectpracticalscoreweight || ' || ' || v\_openedsubjectYn );  end loop;  dbms\_output.put\_line('========================================================================================================================' );  close vcursor;  end;  /  --2. 개설 과목별 성적 조회  --과정명, 과정기간, 강의실명, 과목명, 교사명, 교재명등을 출력하고, 해당 과목을 수강한 모든 교육생의 성적 정보(이름, 주민번호 뒷자리, 필기, 실기)를 같이 출력한다.  create or replace procedure procGetScoreBySubject(  popenedSubjectId number  )  is  cursor vcursor  is  select c.coursename as 과정명  , to\_char(oc.openedstartdate, 'yyyymmdd') as 과정기간\_start  , to\_char(oc.openedcompletiondate, 'yyyymmdd') as 과정기간\_end  , crm.classroomname as 강의실명  , s.subjectname as 과목명  , to\_char(os.subjectStartDate, 'yyyymmdd') as 과목기간\_start  , to\_char(os.subjectEndDate, 'yyyymmdd') as 과목기간\_end  , tb.textbookname as 교재명  , t.teacherName as 교사명  , st.studentName as 학생명  , st.studentResidentNumberBack as 주민번호뒷자리  , g.gradeWrittenScore as 필기점수  , g.gradePracticalScore as 실기점수  , g.gradeAttendanceScore as 출결점수  from tblOpenedCourseoc --개설과정  inner join tblCourse c --강의과정  on oc.courseid = c.courseid  inner join tblOpenedSubjectos --개설과목  on os.openedcourseid = oc.openedcourseid  inner join tblSubject s --과목  on s.subjectid = os.subjectid  inner join tblClassroomcrm --강의실  on crm.classroomid = oc.classroomid  inner join tbltextbook tb --교재  on tb.subjectId = s.subjectId  inner join tblTeacher t --교사  on t.teacherId = oc.teacherid  inner join tblStudentst --학생  on st.studentCourseId = oc.openedcourseid  inner join tblGrade g  on g.studentId = st.studentId  and g.openedSubjectId = os.openedSubjectId  where os.openedSubjectId = popenedSubjectId  order by oc.openedcourseid, os.openedsubjectid  ;  v\_coursenametblCourse.coursename%type;  v\_openedstartdatetblOpenedCourse.openedstartdate%type;  v\_openedcompletiondatetblOpenedCourse.openedcompletiondate%type;  v\_classroomnametblClassroom.classroomname%type;  v\_subjectnametblSubject.subjectname%type;  v\_subjectStartDatetblOpenedSubject.subjectStartDate%type;  v\_subjectEndDatetblOpenedSubject.subjectEndDate%type;  v\_textbooknametbltextbook.textbookname%type;  v\_teacherNametblTeacher.teacherName%type;  v\_studentNametblStudent.studentName%type;  v\_studentResidentNumberBacktblStudent.studentResidentNumberBack%type;  v\_gradeWrittenScoretblGrade.gradeWrittenScore%type;  v\_gradePracticalScoretblGrade.gradePracticalScore%type;  v\_gradeAttendanceScoretblGrade.gradeAttendanceScore%type;  begin  open vcursor;  dbms\_output.put\_line('==============================================================================================================================================');  dbms\_output.put\_line('|| 과정명 || 과정기간 || 강의실명 || 과목명 || 과목기간 || 교재명 || 교사명 || 학생명 || 주민번호뒷자리 || 필기점수 || 실기점수 || 출결점수 ||');  dbms\_output.put\_line('==============================================================================================================================================');  loop  exit when vcursor%notfound;  fetch vcursor into v\_coursename, v\_openedstartdate, v\_openedcompletiondate, v\_classroomname, v\_subjectname, v\_subjectStartDate,  v\_subjectEndDate, v\_textbookname, v\_teacherName, v\_studentName, v\_studentResidentNumberBack, v\_gradeWrittenScore,  v\_gradePracticalScore, v\_gradeAttendanceScore;  dbms\_output.put\_line('|| ' || v\_coursename || ' || ' || to\_char(v\_openedstartdate,'yyyymmdd') || ' ~ ' || to\_char(v\_openedcompletiondate,'yyyymmdd') || ' || ' || v\_classroomname || ' || ' || v\_subjectname || ' || ' || to\_char(v\_subjectStartDate, 'yyyymmdd')  || ' ~ ' || to\_char(v\_subjectEndDate,'yyyymmdd') || ' || ' || v\_textbookname || ' || ' || v\_teacherName || ' || ' || v\_studentName || ' || ' || v\_studentResidentNumberBack || ' || ' || v\_gradeWrittenScore || ' || ' || v\_gradePracticalScore  || ' || ' || v\_gradeAttendanceScore);  end loop;  close vcursor;  dbms\_output.put\_line('==============================================================================================================================================');  end;  /  --3. 교육생별 성적 조회  --교육생 개인별 성적 출력 시 교육생 정보(이름, 주민번호 뒷자리, 과정명, 과정 기간, 강의 실명 등)를 출력하고, 교육생이 수강한 모든 과목에 대한 성적 정보(과목명, 과목 기간, 교사명, 필기, 실기)  create or replace procedure procGetScoreByStudent(  pstudentId number  )  is  cursor vcursor  is  select st.studentName as 학생명  , st.studentResidentNumberBack as 주민번호뒷자리  , c.coursename as 과정명  , to\_char(oc.openedstartdate, 'yyyymmdd') as 과정기간\_start  , to\_char(oc.openedcompletiondate, 'yyyymmdd') as 과정기간\_end  , crm.classroomname as 강의실명  , s.subjectname as 과목명  , to\_char(os.subjectStartDate, 'yyyymmdd') as 과목기간\_start  , to\_char(os.subjectEndDate, 'yyyymmdd') as 과목기간\_end  , t.teacherName as 교사명  , g.gradeWrittenScore as 필기점수  , g.gradePracticalScore as 실기점수  , g.gradeAttendanceScore as 출결점수  from tblOpenedCourseoc --개설과정  inner join tblCourse c --강의과정  on oc.courseid = c.courseid  inner join tblOpenedSubjectos --개설과목  on os.openedcourseid = oc.openedcourseid  inner join tblSubject s --과목  on s.subjectid = os.subjectid  inner join tblClassroomcrm --강의실  on crm.classroomid = oc.classroomid  inner join tblTeacher t --교사  on t.teacherId = oc.teacherid  inner join tblStudentst --학생  on st.studentCourseId = oc.openedcourseid  inner join tblGrade g  on g.studentId = st.studentId  and g.openedSubjectId = os.openedSubjectId  where st.studentId = pstudentId  order by oc.openedcourseid, os.openedsubjectid  ;  v\_coursenametblCourse.coursename%type;  v\_openedstartdatetblOpenedCourse.openedstartdate%type;  v\_openedcompletiondatetblOpenedCourse.openedcompletiondate%type;  v\_classroomnametblClassroom.classroomname%type;  v\_subjectnametblSubject.subjectname%type;  v\_subjectStartDatetblOpenedSubject.subjectStartDate%type;  v\_subjectEndDatetblOpenedSubject.subjectEndDate%type;  v\_teacherNametblTeacher.teacherName%type;  v\_studentNametblStudent.studentName%type;  v\_studentResidentNumberBacktblStudent.studentResidentNumberBack%type;  v\_gradeWrittenScoretblGrade.gradeWrittenScore%type;  v\_gradePracticalScoretblGrade.gradePracticalScore%type;  v\_gradeAttendanceScoretblGrade.gradeAttendanceScore%type;  begin  open vcursor;  dbms\_output.put\_line('==============================================================================================================================================');  dbms\_output.put\_line('|| 학생명 || 주민번호뒷자리 || 과정명 || 과정기간 || 강의실명 || 과목명 || 과목기간 || 교사명 || 필기점수 || 실기점수 || 출결점수 ||');  dbms\_output.put\_line('==============================================================================================================================================');  loop  exit when vcursor%notfound;  fetch vcursor into v\_studentName, v\_studentResidentNumberBack, v\_coursename, v\_openedstartdate, v\_openedcompletiondate, v\_classroomname, v\_subjectname,  v\_subjectStartDate, v\_subjectEndDate, v\_teacherName, v\_gradeWrittenScore, v\_gradePracticalScore, v\_gradeAttendanceScore;  dbms\_output.put\_line('|| ' || v\_studentName || ' || ' || v\_studentResidentNumberBack || ' || ' || v\_coursename || ' || ' || v\_openedstartdate || ' ~ ' || v\_openedcompletiondate || ' || ' || v\_classroomname  || ' || ' || v\_subjectname || ' || ' || v\_subjectStartDate || ' ~ ' || v\_subjectEndDate || ' || ' || v\_teacherName || ' || ' || v\_gradeWrittenScore || ' || ' || v\_gradePracticalScore || ' || ' || v\_gradeAttendanceScore);  end loop;  close vcursor;  dbms\_output.put\_line('==============================================================================================================================================');  end;  / | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 요구사항번호 | A-8. 출결 관리 및 출결 조회 | 작성일 | 2025-02-07 |
| 요구사항명 | 출결 관리 및 출결 조회 | 작성자 | 김원중 |
| DML | | | |
| -출결 관리  \*/  /  set serveroutputon;  /  /\* 출결 \*/  --1.기간별 조회  CREATE OR REPLACE PROCEDURE procGetAttendanceByDateRange(  pStartDate IN DATE,  pEndDate IN DATE  )  IS  BEGIN  FOR rec IN (  SELECT  s.studentName AS 학생이름,  TO\_CHAR(a.attendanceDate, 'yyyy-mm-dd') AS 날짜,  a.attendanceCheckInTime AS 입실시간,  a.attendanceCheckOutTime AS 퇴실시간,  a.attendanceStatus AS 출석상태  FROM tblAttendance a  JOIN tblStudent s ON a.studentId = s.studentId  WHERE a.attendanceDate BETWEEN pStartDate AND pEndDate  ) LOOP  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('학생이름: ' || rec.학생이름);  DBMS\_OUTPUT.PUT\_LINE('날짜: ' || rec.날짜);  DBMS\_OUTPUT.PUT\_LINE('입실시간: ' || rec.입실시간);  DBMS\_OUTPUT.PUT\_LINE('퇴실시간: ' || rec.퇴실시간);  DBMS\_OUTPUT.PUT\_LINE('출석상태: ' || rec.출석상태);  DBMS\_OUTPUT.PUT\_LINE('======================================================');  END LOOP;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('오류 발생: ' || SQLERRM);  END;  /  BEGIN  procGetAttendanceByDateRange('2024-12-01','2024-12-31');  END;  /  --2. 월조회  CREATE OR REPLACE PROCEDURE procGetAttendanceByMonth(  pYear IN NUMBER,  pMonth IN NUMBER  )  IS  BEGIN  FOR rec IN (  SELECT  s.studentName AS 학생이름,  TO\_CHAR(a.attendanceDate, 'yyyy-mm-dd') AS 날짜,  a.attendanceCheckInTime AS 입실시간,  a.attendanceCheckOutTime AS 퇴실시간,  a.attendanceStatus AS 출석상태  FROM tblAttendance a  JOIN tblStudent s ON a.studentId = s.studentId  WHERE EXTRACT(YEAR FROM a.attendanceDate) = pYear  AND EXTRACT(MONTH FROM a.attendanceDate) = pMonth  ) LOOP  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('학생이름: ' || rec.학생이름);  DBMS\_OUTPUT.PUT\_LINE('날짜: ' || rec.날짜);  DBMS\_OUTPUT.PUT\_LINE('입실시간: ' || rec.입실시간);  DBMS\_OUTPUT.PUT\_LINE('퇴실시간: ' || rec.퇴실시간);  DBMS\_OUTPUT.PUT\_LINE('출석상태: ' || rec.출석상태);  DBMS\_OUTPUT.PUT\_LINE('======================================================');  END LOOP;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('오류 발생: ' || SQLERRM);  END;  /  BEGIN  procGetAttendanceByMonth('2024','12');  END;  /  --3.날짜 지정 조회  CREATE OR REPLACE PROCEDURE procGetAttendanceByDate(  pAttendanceDate IN DATE  )  IS  BEGIN  FOR rec IN (  SELECT  s.studentName AS 학생이름,  TO\_CHAR(a.attendanceDate, 'yyyy-mm-dd') AS 날짜,  a.attendanceCheckInTime AS 입실시간,  a.attendanceCheckOutTime AS 퇴실시간,  a.attendanceStatus AS 출석상태  FROM tblAttendance a  JOIN tblStudent s ON a.studentId = s.studentId  WHERE a.attendanceDate = pAttendanceDate  ) LOOP  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('학생이름: ' || rec.학생이름);  DBMS\_OUTPUT.PUT\_LINE('날짜: ' || rec.날짜);  DBMS\_OUTPUT.PUT\_LINE('입실시간: ' || rec.입실시간);  DBMS\_OUTPUT.PUT\_LINE('퇴실시간: ' || rec.퇴실시간);  DBMS\_OUTPUT.PUT\_LINE('출석상태: ' || rec.출석상태);  DBMS\_OUTPUT.PUT\_LINE('======================================================');  END LOOP;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('오류 발생: ' || SQLERRM);  END;  /  BEGIN  procGetAttendanceByDate('2024-12-24');  END;  /  --특정 학생 조회  CREATE OR REPLACE PROCEDURE procGetAttendanceByStudent(  pStudentName IN VARCHAR2  )  IS  BEGIN  FOR rec IN (  SELECT  s.studentName AS 학생이름,  TO\_CHAR(a.attendanceDate, 'yyyy-mm-dd') AS 날짜,  a.attendanceCheckInTime AS 입실시간,  a.attendanceCheckOutTime AS 퇴실시간,  a.attendanceStatus AS 출석상태  FROM tblAttendance a  JOIN tblStudent s ON a.studentId = s.studentId  WHERE s.studentName = pStudentName  ) LOOP  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('학생이름: ' || rec.학생이름);  DBMS\_OUTPUT.PUT\_LINE('날짜: ' || rec.날짜);  DBMS\_OUTPUT.PUT\_LINE('입실시간: ' || rec.입실시간);  DBMS\_OUTPUT.PUT\_LINE('퇴실시간: ' || rec.퇴실시간);  DBMS\_OUTPUT.PUT\_LINE('출석상태: ' || rec.출석상태);  DBMS\_OUTPUT.PUT\_LINE('======================================================');  END LOOP;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('오류 발생: ' || SQLERRM);  END;  /  BEGIN  procGetAttendanceByStudent('김민지');  END;  /  --특정일 지각자  CREATE OR REPLACE PROCEDURE procGetLateAttendanceByDate(  pAttendanceDate IN DATE  )  IS  BEGIN  FOR rec IN (  SELECT  s.studentName AS 학생이름,  TO\_CHAR(a.attendanceDate, 'yyyy-mm-dd') AS 날짜,  a.attendanceCheckInTime AS 입실시간,  a.attendanceCheckOutTime AS 퇴실시간,  a.attendanceStatus AS 출석상태  FROM tblAttendance a  JOIN tblStudent s ON a.studentId = s.studentId  WHERE a.attendanceStatus = '지각'  AND a.attendanceDate = pAttendanceDate  ) LOOP  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('학생이름: ' || rec.학생이름);  DBMS\_OUTPUT.PUT\_LINE('날짜: ' || rec.날짜);  DBMS\_OUTPUT.PUT\_LINE('입실시간: ' || rec.입실시간);  DBMS\_OUTPUT.PUT\_LINE('퇴실시간: ' || rec.퇴실시간);  DBMS\_OUTPUT.PUT\_LINE('출석상태: ' || rec.출석상태);  DBMS\_OUTPUT.PUT\_LINE('======================================================');  END LOOP;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('오류 발생: ' || SQLERRM);  END;  /  BEGIN  procGetLateAttendanceByDate('2024-12-24');  END;  / | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 요구사항번호 | A-9. 기물 관리 | 작성일 | 2025-02-07 |
| 요구사항명 | 기물 관리 | 작성자 | 김원중 |
| DML | | | |
| /\* 기물 \*/  --기물 추가  CREATE OR REPLACE PROCEDURE procEquipment(  pequipmentId IN NUMBER,  pitemName IN VARCHAR2,  pquantity IN NUMBER  )  IS  vequipmentIdtblEquipment.equipmentId%TYPE;  vpitemNametblEquipment.itemName%TYPE;  vquantitytblEquipment.quantity%TYPE;  BEGIN  INSERT INTO tblEquipment (equipmentId, itemName, quantity)  VALUES ((SELECT NVL(MAX(equipmentId), 0) + 1 FROM tblEquipment), pitemName, pquantity);  SELECT MAX(equipmentId) INTO vequipmentId FROM tblEquipment;  SELECT equipmentId, itemName, quantity  INTO vequipmentId, vpitemName, vquantity  FROM tblEquipment  WHERE equipmentId = vequipmentId;  DBMS\_OUTPUT.PUT\_LINE('기물명 : ' || vpitemName || ' 수량 : ' || vquantity);  --DBMS\_OUTPUT.PUT\_LINE(vpitemNam || vquantity);  END;  /  BEGIN  procEquipment(7,'스텐바이미',50);  END;  /  --기물 삭제  CREATE OR REPLACE PROCEDURE procEquipment(  pequipmentId NUMBER  )  IS  vcountNUMBER;  BEGIN  SELECT COUNT(\*) INTO vcount  FROM tblEquipment  WHERE equipmentId = pequipmentId;  IF vcount> 0 THEN  DELETE FROM tblEquipment  WHERE equipmentId = pequipmentId;  COMMIT;  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('기물이 삭제되었습니다. (ID: ' || pequipmentId || ')');  DBMS\_OUTPUT.PUT\_LINE('======================================================');  ELSE  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('삭제할 기물이 없습니다. (ID: ' || pequipmentId || ')');  DBMS\_OUTPUT.PUT\_LINE('======================================================');  END IF;  EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  DBMS\_OUTPUT.PUT\_LINE('오류 발생: ' || SQLERRM);  END;  /  BEGIN  procEquipment(1);  END;  /  --기물 수정  CREATE OR REPLACE PROCEDURE procUpdateEquipment(  pequipmentId IN NUMBER,  pitemName IN VARCHAR2 DEFAULT NULL,  pquantity IN NUMBER DEFAULT NULL  )  IS  vcountNUMBER;  vcurrentItemNametblEquipment.itemName%TYPE;  vcurrentQuantitytblEquipment.quantity%TYPE;  BEGIN  SELECT COUNT(\*) INTO vcount  FROM tblEquipment  WHERE equipmentId = pequipmentId;  IF vcount = 0 THEN  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('수정할 기물이 없습니다. (ID: ' || pequipmentId || ')');  DBMS\_OUTPUT.PUT\_LINE('======================================================');  RETURN;  END IF;  SELECT itemName, quantity  INTO vcurrentItemName, vcurrentQuantity  FROM tblEquipment  WHERE equipmentId = pequipmentId;  UPDATE tblEquipment  SET itemName = NVL(pitemName, vcurrentItemName),  quantity = NVL(pquantity, vcurrentQuantity)  WHERE equipmentId = pequipmentId;  COMMIT;  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('기물이 수정되었습니다. (ID: ' || pequipmentId || ')');  DBMS\_OUTPUT.PUT\_LINE('기물명: ' || NVL(pitemName, vcurrentItemName));  DBMS\_OUTPUT.PUT\_LINE('수량: ' || NVL(pquantity, vcurrentQuantity));  DBMS\_OUTPUT.PUT\_LINE('======================================================');  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('기물을 찾을 수 없습니다. (ID: ' || pequipmentId || ')');  WHEN OTHERS THEN  ROLLBACK;  DBMS\_OUTPUT.PUT\_LINE('오류 발생: ' || SQLERRM);  END;  /  BEGIN  procEquipment(7,'스텐바이미',12);  END;  /  --기물 조회  CREATE OR REPLACE PROCEDURE procSelectEquipment(  pequipmentId IN NUMBER DEFAULT NULL  )  IS  vitemNametblEquipment.itemName%TYPE;  vquantitytblEquipment.quantity%TYPE;  CURSOR curEquipment IS  SELECT equipmentId, itemName, quantity  FROM tblEquipment;  BEGIN  IF pequipmentId IS NOT NULL THEN  BEGIN  SELECT itemName, quantity  INTO vitemName, vquantity  FROM tblEquipment  WHERE equipmentId = pequipmentId;  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('기물 조회 결과 (ID: ' || pequipmentId || ')');  DBMS\_OUTPUT.PUT\_LINE('기물명: ' || vitemName);  DBMS\_OUTPUT.PUT\_LINE('수량: ' || vquantity);  DBMS\_OUTPUT.PUT\_LINE('======================================================');  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('해당 기물이 없습니다. (ID: ' || pequipmentId || ')');  END;  ELSE  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('전체 기물 목록');  DBMS\_OUTPUT.PUT\_LINE('======================================================');  FOR rec IN curEquipment LOOP  DBMS\_OUTPUT.PUT\_LINE('ID: ' || rec.equipmentId || ' | 기물명: ' || rec.itemName || ' | 수량: ' || rec.quantity);  END LOOP;  DBMS\_OUTPUT.PUT\_LINE('======================================================');  END IF;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('오류 발생: ' || SQLERRM);  END;  /  BEGIN  procSelectEquipment();  END;  / | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 요구사항번호 | A-10. 국민 취업 제도 참여여부 조회 | 작성일 | 2025-02-07 |
| 요구사항명 | 국민 취업 제도 참여여부 조회 | 작성자 | 김원중 |
| DML | | | |
| /\* 국취제 참여여부 조회\*/  -- 학생의 국취제 참여 여부 조회  CREATE OR REPLACE PROCEDURE procGetNationalRetrievalParticipation IS  BEGIN  FOR rec IN (SELECT s.studentName AS 학생이름,  n.nationalRetrievalParticipation AS 참여여부  FROM tblNationalRetrieval n  JOIN tblStudent s ON n.studentId = s.studentId) LOOP  DBMS\_OUTPUT.PUT\_LINE('학생이름: ' || rec.학생이름 || ', 참여여부: ' || rec.참여여부);  END LOOP;  END;  /  BEGIN  procGetNationalRetrievalParticipation ();  END;  /  -- 국취제 지원 유형 조회  CREATE OR REPLACE PROCEDURE procGetSupportType IS  BEGIN  FOR rec IN (SELECT s.studentName AS 학생이름,  n.nationalRetrievalSupport || '유형' AS 지원유형  FROM tblNationalRetrieval n  JOIN tblStudent s ON n.studentId = s.studentId) LOOP  DBMS\_OUTPUT.PUT\_LINE('학생이름: ' || rec.학생이름 || ', 지원유형: ' || rec.지원유형);  END LOOP;  END;  /  BEGIN  procGetSupportType();  END;  /  -- 국취제 1유형 지원 학생 조회  CREATE OR REPLACE PROCEDURE procGetSupportType1 IS  BEGIN  FOR rec IN (SELECT s.studentName AS "1유형 지원학생"  FROM tblNationalRetrieval n  JOIN tblStudent s ON n.studentId = s.studentId  WHERE n.nationalRetrievalSupport = '1') LOOP  DBMS\_OUTPUT.PUT\_LINE('1유형 지원학생: ' || rec."1유형 지원학생");  END LOOP;  END;  /  BEGIN  procGetSupportType1();  END;  /  -- 국취제 2유형 지원 학생 조회  CREATE OR REPLACE PROCEDURE procGetSupportType2 IS  BEGIN  FOR rec IN (SELECT s.studentName AS "2유형 지원학생"  FROM tblNationalRetrieval n  JOIN tblStudent s ON n.studentId = s.studentId  WHERE n.nationalRetrievalSupport = '2') LOOP  DBMS\_OUTPUT.PUT\_LINE('2유형 지원학생: ' || rec."2유형 지원학생");  END LOOP;  END;  /  BEGIN  procGetSupportType2();  END;  /  -- 국취제 신청일 조회  CREATE OR REPLACE PROCEDURE procGetApplicationDate IS  BEGIN  FOR rec IN (SELECT s.studentName AS 학생이름,  TO\_CHAR(n.ntnlRtrvlplctnDt, 'yyyy-mm-dd') AS 신청일  FROM tblNationalRetrieval n  JOIN tblStudent s ON n.studentId = s.studentId) LOOP  DBMS\_OUTPUT.PUT\_LINE('학생이름: ' || rec.학생이름 || ', 신청일: ' || rec.신청일);  END LOOP;  END;  /  BEGIN  procGetApplicationDate();  END;  /  -- 국취제 승인여부 조회  CREATE OR REPLACE PROCEDURE procGetApprovalStatus IS  BEGIN  FOR rec IN (SELECT s.studentName AS 학생이름,  n.ntnlRtrvlprvlSts AS 승인여부  FROM tblNationalRetrieval n  JOIN tblStudent s ON n.studentId = s.studentId) LOOP  DBMS\_OUTPUT.PUT\_LINE('학생이름: ' || rec.학생이름 || ', 승인여부: ' || rec.승인여부);  END LOOP;  END;  /  BEGIN  procGetApprovalStatus();  END;  /  -- 국취제승인완료된 학생 조회  CREATE OR REPLACE PROCEDURE procGetApprovedStudents IS  BEGIN  FOR rec IN (SELECT s.studentName AS 승인완료된학생  FROM tblNationalRetrieval n  JOIN tblStudent s ON n.studentId = s.studentId  WHERE n.ntnlRtrvlprvlSts = '승인완료') LOOP  DBMS\_OUTPUT.PUT\_LINE('승인완료된학생: ' || rec.승인완료된학생);  END LOOP;  END;  /  BEGIN  procGetApprovedStudents();  END;  /  -- 국취제승인거부된 학생 조회  CREATE OR REPLACE PROCEDURE procGetRejectedStudents IS  BEGIN  FOR rec IN (SELECT s.studentName AS 승인거부된학생  FROM tblNationalRetrieval n  JOIN tblStudent s ON n.studentId = s.studentId  WHERE n.ntnlRtrvlprvlSts = '승인거부') LOOP  DBMS\_OUTPUT.PUT\_LINE('승인거부된학생: ' || rec.승인거부된학생);  END LOOP;  END;  /  BEGIN  procGetRejectedStudents();  END;  / | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 요구사항번호 | A-11. 공지사항 관리 | 작성일 | 2025-02-07 |
| 요구사항명 | 공지사항 관리 | 작성자 | 김원중 |
| DML | | | |
| /\* 공지사항 \*/  --공지사항 등록  -- SEQUENCE 생성  CREATE SEQUENCE NOTICE\_SEQ  START WITH 1  INCREMENT BY 1  NOCACHE  NOCYCLE;  /  -- 공지사항 추가  CREATE OR REPLACE PROCEDURE procInputNotice(  pnoticeTitle IN VARCHAR2,  pnoticeContent IN VARCHAR2,  padminManagerId IN NUMBER  )  IS  vnoticeIdtblNotice.noticeId%TYPE;  BEGIN  -- SEQUENCE를 사용하여 공지 ID 생성  vnoticeId := NOTICE\_SEQ.NEXTVAL;  -- 현재 날짜는 SYSDATE로 자동 설정  INSERT INTO tblNotice(noticeId, noticeTitle, noticeContent, noticeCreationDate, adminManagerId)  VALUES(vnoticeId, pnoticeTitle, pnoticeContent, SYSDATE, padminManagerId);  -- 공지사항 등록 성공 메시지 출력  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('공지사항이 등록되었습니다.');  DBMS\_OUTPUT.PUT\_LINE('공지 ID : ' || vnoticeId);  DBMS\_OUTPUT.PUT\_LINE('제목 : ' || pnoticeTitle);  DBMS\_OUTPUT.PUT\_LINE('내용 : ' || pnoticeContent);  DBMS\_OUTPUT.PUT\_LINE('날짜 : ' || TO\_CHAR(SYSDATE, 'YYYY-MM-DD'));  DBMS\_OUTPUT.PUT\_LINE('관리자 ID : ' || padminManagerId);  DBMS\_OUTPUT.PUT\_LINE('======================================================');  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('오류 발생: ' || SQLERRM);  END;  /  BEGIN  procInputNotice('관리자11111', '테스트중~~', 3);  END;  /  --공지사항 조회  CREATE OR REPLACE PROCEDURE procGetNotice(  pnoticeId IN NUMBER DEFAULT NULL  )  IS  BEGIN  IF pnoticeId IS NOT NULL THEN  DECLARE  vnoticeTitletblNotice.noticeTitle%TYPE;  vnoticeContenttblNotice.noticeContent%TYPE;  vnoticeCreationDatetblNotice.noticeCreationDate%TYPE;  vadminManagerIdtblNotice.adminManagerId%TYPE;  BEGIN  SELECT noticeTitle, noticeContent, noticeCreationDate, adminManagerId  INTO vnoticeTitle, vnoticeContent, vnoticeCreationDate, vadminManagerId  FROM tblNotice  WHERE noticeId = pnoticeId;  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('공지사항 조회 (ID: ' || pnoticeId || ')');  DBMS\_OUTPUT.PUT\_LINE('제목: ' || vnoticeTitle);  DBMS\_OUTPUT.PUT\_LINE('내용: ' || vnoticeContent);  DBMS\_OUTPUT.PUT\_LINE('날짜: ' || TO\_CHAR(vnoticeCreationDate, 'YYYY-MM-DD'));  DBMS\_OUTPUT.PUT\_LINE('관리자 ID: ' || vadminManagerId);  DBMS\_OUTPUT.PUT\_LINE('======================================================');  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('해당 ID의 공지사항이 없습니다. (ID: ' || pnoticeId || ')');  END;  ELSE  FOR rec IN (SELECT noticeId, noticeTitle, noticeContent, noticeCreationDate, adminManagerId  FROM tblNotice ORDER BY noticeCreationDateasc)  LOOP  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('공지사항 ID: ' || rec.noticeId);  DBMS\_OUTPUT.PUT\_LINE('제목: ' || rec.noticeTitle);  DBMS\_OUTPUT.PUT\_LINE('내용: ' || rec.noticeContent);  DBMS\_OUTPUT.PUT\_LINE('날짜: ' || TO\_CHAR(rec.noticeCreationDate, 'YYYY-MM-DD'));  DBMS\_OUTPUT.PUT\_LINE('관리자 ID: ' || rec.adminManagerId);  DBMS\_OUTPUT.PUT\_LINE('======================================================');  END LOOP;  END IF;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('오류 발생: ' || SQLERRM);  END;  /  BEGIN  procGetNotice();  END;  /  --공지사항 수정  CREATE OR REPLACE PROCEDURE procUpdateNotice(  pnoticeId IN NUMBER,  pnoticeTitle IN VARCHAR2 DEFAULT NULL,  pnoticeContent IN VARCHAR2 DEFAULT NULL,  pnoticeCreationDate IN DATE DEFAULT NULL  )  IS  vcountNUMBER;  vcurrentTitletblNotice.noticeTitle%TYPE;  vcurrentContenttblNotice.noticeContent%TYPE;  vcurrentDatetblNotice.noticeCreationDate%TYPE;  BEGIN  SELECT COUNT(\*) INTO vcount  FROM tblNotice  WHERE noticeId = pnoticeId;  IF vcount = 0 THEN  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('수정할 공지사항이 없습니다. (ID: ' || pnoticeId || ')');  DBMS\_OUTPUT.PUT\_LINE('======================================================');  RETURN;  END IF;  SELECT noticeTitle, noticeContent, noticeCreationDate  INTO vcurrentTitle, vcurrentContent, vcurrentDate  FROM tblNotice  WHERE noticeId = pnoticeId;  UPDATE tblNotice  SET noticeTitle = NVL(pnoticeTitle, vcurrentTitle),  noticeContent = NVL(pnoticeContent, vcurrentContent),  noticeCreationDate = NVL(pnoticeCreationDate, vcurrentDate)  WHERE noticeId = pnoticeId;  COMMIT;  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('공지사항이 수정되었습니다. (ID: ' || pnoticeId || ')');  DBMS\_OUTPUT.PUT\_LINE('제목: ' || NVL(pnoticeTitle, vcurrentTitle));  DBMS\_OUTPUT.PUT\_LINE('내용: ' || NVL(pnoticeContent, vcurrentContent));  DBMS\_OUTPUT.PUT\_LINE('날짜: ' || TO\_CHAR(NVL(pnoticeCreationDate, vcurrentDate), 'YYYY-MM-DD'));  DBMS\_OUTPUT.PUT\_LINE('======================================================');  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('공지사항을 찾을 수 없습니다. (ID: ' || pnoticeId || ')');  WHEN OTHERS THEN  ROLLBACK;  DBMS\_OUTPUT.PUT\_LINE('오류 발생: ' || SQLERRM);  END;  /  BEGIN  procUpdateNotice('13','공휴일 휴강안내~','','');  END;  /  --공지사항 삭제  CREATE OR REPLACE PROCEDURE procDeleteNotice(  pnoticeId IN NUMBER  )  IS  vcountNUMBER;  BEGIN  SELECT COUNT(\*) INTO vcount  FROM tblNotice  WHERE noticeId = pnoticeId;  IF vcount> 0 THEN  DELETE FROM tblNotice  WHERE noticeId = pnoticeId;  COMMIT;  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('공지사항이 삭제되었습니다. (ID: ' || pnoticeId || ')');  DBMS\_OUTPUT.PUT\_LINE('======================================================');  ELSE  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('삭제할 공지사항이 없습니다. (ID: ' || pnoticeId || ')');  DBMS\_OUTPUT.PUT\_LINE('======================================================');  END IF;  EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  DBMS\_OUTPUT.PUT\_LINE('오류 발생: ' || SQLERRM);  END;  /  BEGIN  procDeleteNotice('13');  END;  / | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| 요구사항번호 | A-12. 자료실, 자율게시판 관리 | 작성일 | 2025-02-07 |
| 요구사항명 | 자료실, 자율게시판 관리 | 작성자 | 김원중 |
| DML | | | |
| /\* 자료실 관리 \*/  /\*  --자료실 등록  CREATE SEQUENCE ResourceRoom\_SEQ  START WITH 1  INCREMENT BY 1  NOCACHE  NOCYCLE;  CREATE OR REPLACE PROCEDURE procInputResourceRoom(  presourceTitle IN VARCHAR2,  presourceContent IN VARCHAR2,  presourceCreationDate IN DATE,  pteacherId IN NUMBER  )  IS  vresourceRoomIdtblResourceRoom.resourceRoomId%TYPE;  BEGIN  vresourceRoomId := notice\_seq.NEXTVAL;  INSERT INTO tblResourceRoom(resourceRoomId, resourceTitle, resourceContent, resourceCreationDate,teacherId)  VALUES(vresourceRoomId, presourceTitle, presourceContent, presourceCreationDate, pteacherId);  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('자료실 게시물이 등록되었습니다.');  DBMS\_OUTPUT.PUT\_LINE('제목 : ' || presourceTitle);  DBMS\_OUTPUT.PUT\_LINE('내용 : ' || presourceContent);  DBMS\_OUTPUT.PUT\_LINE('날짜 : ' || TO\_CHAR(presourceCreationDate, 'YYYY-MM-DD'));  DBMS\_OUTPUT.PUT\_LINE('교사 ID : ' || pteacherId);  DBMS\_OUTPUT.PUT\_LINE('======================================================');  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('오류 발생: ' || SQLERRM);  END;  \*/  /  --자료실 조회  CREATE OR REPLACE PROCEDURE procGetResourceRoom(  presourceRoomId IN NUMBER DEFAULT NULL  )  IS  BEGIN  IF presourceRoomId IS NOT NULL THEN  DECLARE  vresourceTitletblResourceRoom.resourceTitle%TYPE;  vresourceContenttblResourceRoom.resourceContent%TYPE;  vresourceCreationDatetblResourceRoom.resourceCreationDate%TYPE;  vteacherIdtblResourceRoom.teacherId%TYPE;  BEGIN  SELECT resourceTitle, resourceContent, resourceCreationDate, teacherId  INTO vresourceTitle, vresourceContent, vresourceCreationDate, vteacherId  FROM tblResourceRoom  WHERE resourceRoomId = presourceRoomId;  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('자료실 조회 (ID: ' || presourceRoomId || ')');  DBMS\_OUTPUT.PUT\_LINE('제목: ' || vresourceTitle);  DBMS\_OUTPUT.PUT\_LINE('내용: ' || vresourceContent);  DBMS\_OUTPUT.PUT\_LINE('날짜: ' || TO\_CHAR(vresourceCreationDate, 'YYYY-MM-DD'));  DBMS\_OUTPUT.PUT\_LINE('교사 ID: ' || vteacherId);  DBMS\_OUTPUT.PUT\_LINE('======================================================');  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('해당 ID의 자료실 게시물이 없습니다. (ID: ' || presourceRoomId || ')');  END;  ELSE  FOR rec IN (SELECT resourceRoomId, resourceTitle, resourceContent, resourceCreationDate, teacherId  FROM tblResourceRoom ORDER BY resourceCreationDate DESC)  LOOP  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('자료실 ID: ' || rec.resourceRoomId);  DBMS\_OUTPUT.PUT\_LINE('제목: ' || rec.resourceTitle);  DBMS\_OUTPUT.PUT\_LINE('내용: ' || rec.resourceContent);  DBMS\_OUTPUT.PUT\_LINE('날짜: ' || TO\_CHAR(rec.resourceCreationDate, 'YYYY-MM-DD'));  DBMS\_OUTPUT.PUT\_LINE('교사 ID: ' || rec.teacherId);  DBMS\_OUTPUT.PUT\_LINE('======================================================');  END LOOP;  END IF;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('오류 발생: ' || SQLERRM);  END;  /  /  --전체조회  BEGIN  procGetResourceRoom('');  END;  /  /  --1번 조회  BEGIN  procGetResourceRoom('1');  END;  /  --자료실 수정  CREATE OR REPLACE PROCEDURE procUpdateResourceRoom(  presourceRoomId IN NUMBER,  presourceTitle IN VARCHAR2 DEFAULT NULL,  presourceContent IN VARCHAR2 DEFAULT NULL,  presourceCreationDate IN DATE DEFAULT NULL  )  IS  vcountNUMBER;  vcurrentTitletblResourceRoom.resourceTitle%TYPE;  vcurrentContenttblResourceRoom.resourceContent%TYPE;  vcurrentDatetblResourceRoom.resourceCreationDate%TYPE;  BEGIN  SELECT COUNT(\*) INTO vcount  FROM tblResourceRoom  WHERE resourceRoomId = presourceRoomId;  IF vcount = 0 THEN  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('수정할 자료실 게시물이 없습니다. (ID: ' || presourceRoomId || ')');  DBMS\_OUTPUT.PUT\_LINE('======================================================');  RETURN;  END IF;  SELECT resourceTitle, resourceContent, resourceCreationDate  INTO vcurrentTitle, vcurrentContent, vcurrentDate  FROM tblResourceRoom  WHERE resourceRoomId = presourceRoomId;  UPDATE tblResourceRoom  SET resourceTitle = NVL(presourceTitle, vcurrentTitle),  resourceContent = NVL(presourceContent, vcurrentContent),  resourceCreationDate = NVL(presourceCreationDate, vcurrentDate)  WHERE resourceRoomId = presourceRoomId;  COMMIT;  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('자료실 게시물의 내용이 수정되었습니다. (ID: ' || presourceRoomId || ')');  DBMS\_OUTPUT.PUT\_LINE('제목: ' || NVL(presourceTitle, vcurrentTitle));  DBMS\_OUTPUT.PUT\_LINE('내용: ' || NVL(presourceContent, vcurrentContent));  DBMS\_OUTPUT.PUT\_LINE('날짜: ' || TO\_CHAR(NVL(presourceCreationDate, vcurrentDate), 'YYYY-MM-DD'));  DBMS\_OUTPUT.PUT\_LINE('======================================================');  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('자료실 게시물을 찾을 수 없습니다. (ID: ' || presourceRoomId || ')');  WHEN OTHERS THEN  ROLLBACK;  DBMS\_OUTPUT.PUT\_LINE('오류 발생: ' || SQLERRM);  END;  /  BEGIN  procUpdateResourceRoom('1','관리자','테스트중','2025-12-12');  END;  /  --자료실 삭제  CREATE OR REPLACE PROCEDURE procDeleteResourceRoom(  presourceRoomId IN NUMBER  )  IS  vcountNUMBER;  BEGIN  SELECT COUNT(\*) INTO vcount  FROM tblResourceRoom  WHERE resourceRoomId = presourceRoomId;  IF vcount> 0 THEN  DELETE FROM tblResourceRoom  WHERE resourceRoomId = presourceRoomId;  COMMIT;  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('자료실 게시물이 삭제되었습니다. (ID: ' || presourceRoomId || ')');  DBMS\_OUTPUT.PUT\_LINE('======================================================');  ELSE  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('삭제할 자료실 게시물이 없습니다. (ID: ' || presourceRoomId || ')');  DBMS\_OUTPUT.PUT\_LINE('======================================================');  END IF;  EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  DBMS\_OUTPUT.PUT\_LINE('오류 발생: ' || SQLERRM);  END;  /  BEGIN  procDeleteResourceRoom('1');  END;  /  /\* 자율게시판 관리\*/  --자율게시판 조회  CREATE OR REPLACE PROCEDURE procGetFreeBoard(  pfreeBoardId IN NUMBER DEFAULT NULL  )  IS  BEGIN  IF pfreeBoardId IS NOT NULL THEN  DECLARE  vfreeBoardTitletblFreeBoard.freeBoardTitle%TYPE;  vfreeBoardContenttblFreeBoard.freeBoardContent%TYPE;  vfreeBoardCreationDatetblFreeBoard.freeBoardCreationDate%TYPE;  vstudentIdtblFreeBoard.studentId%TYPE;  BEGIN  SELECT freeBoardTitle, freeBoardContent, freeBoardCreationDate, studentId  INTO vfreeBoardTitle, vfreeBoardContent, vfreeBoardCreationDate, vstudentId  FROM tblFreeBoard  WHERE freeBoardId = pfreeBoardId;  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('자율게시판 조회 (ID: ' || pfreeBoardId || ')');  DBMS\_OUTPUT.PUT\_LINE('제목: ' || vfreeBoardTitle);  DBMS\_OUTPUT.PUT\_LINE('내용: ' || vfreeBoardContent);  DBMS\_OUTPUT.PUT\_LINE('날짜: ' || TO\_CHAR(vfreeBoardCreationDate, 'YYYY-MM-DD'));  DBMS\_OUTPUT.PUT\_LINE('학생 ID: ' || vstudentId);  DBMS\_OUTPUT.PUT\_LINE('======================================================');  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('해당 ID의 게시물이 없습니다. (ID: ' || pfreeBoardId || ')');  END;  ELSE  -- 게시물 ID가 없는 경우 전체 게시물을 조회  FOR rec IN (SELECT freeBoardId, freeBoardTitle, freeBoardContent, freeBoardCreationDate, studentId  FROM tblFreeBoard ORDER BY freeBoardCreationDate DESC)  LOOP  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('자율게시판 ID: ' || rec.freeBoardId);  DBMS\_OUTPUT.PUT\_LINE('제목: ' || rec.freeBoardTitle);  DBMS\_OUTPUT.PUT\_LINE('내용: ' || rec.freeBoardContent);  DBMS\_OUTPUT.PUT\_LINE('날짜: ' || TO\_CHAR(rec.freeBoardCreationDate, 'YYYY-MM-DD'));  DBMS\_OUTPUT.PUT\_LINE('학생 ID: ' || rec.studentId);  DBMS\_OUTPUT.PUT\_LINE('======================================================');  END LOOP;  END IF;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('오류 발생: ' || SQLERRM);  END;  /  --3번 게시물 조회  BEGIN  procGetFreeBoard('3');  END;  /  --전체 조회  BEGIN  procGetFreeBoard('');  END;  /  --자율게시판 수정  CREATE OR REPLACE PROCEDURE procUpdateFreeBoard(  pfreeBoardId IN NUMBER,  pfreeBoardTitle IN VARCHAR2 DEFAULT NULL,  pfreeBoardContent IN VARCHAR2 DEFAULT NULL  )  IS  vcountNUMBER;  vcurrentTitletblFreeBoard.freeBoardTitle%TYPE;  vcurrentContenttblFreeBoard.freeBoardContent%TYPE;  BEGIN  -- 게시물 존재 여부 확인  SELECT COUNT(\*) INTO vcount  FROM tblFreeBoard  WHERE freeBoardId = pfreeBoardId;  IF vcount = 0 THEN  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('수정할 게시물이 없습니다. (ID: ' || pfreeBoardId || ')');  DBMS\_OUTPUT.PUT\_LINE('======================================================');  RETURN;  END IF;  SELECT freeBoardTitle, freeBoardContent  INTO vcurrentTitle, vcurrentContent  FROM tblFreeBoard  WHERE freeBoardId = pfreeBoardId;  UPDATE tblFreeBoard  SET freeBoardTitle = NVL(pfreeBoardTitle, vcurrentTitle),  freeBoardContent = NVL(pfreeBoardContent, vcurrentContent)  WHERE freeBoardId = pfreeBoardId;  COMMIT;  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('게시물 내용이 수정되었습니다. (ID: ' || pfreeBoardId || ')');  DBMS\_OUTPUT.PUT\_LINE('제목: ' || NVL(pfreeBoardTitle, vcurrentTitle));  DBMS\_OUTPUT.PUT\_LINE('내용: ' || NVL(pfreeBoardContent, vcurrentContent));  DBMS\_OUTPUT.PUT\_LINE('======================================================');  EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  DBMS\_OUTPUT.PUT\_LINE('오류 발생: ' || SQLERRM);  END;  /  BEGIN  procUpdateFreeBoard('1','관리자','관리자에 의해 게시물 내용이 수정되었습니다.');  END;  /  --자율게시판 삭제  CREATE OR REPLACE PROCEDURE procDeleteFreeBoard(  pfreeBoardId IN NUMBER  )  IS  vcountNUMBER;  BEGIN  -- 게시물 존재 여부 확인  SELECT COUNT(\*) INTO vcount  FROM tblFreeBoard  WHERE freeBoardId = pfreeBoardId;  IF vcount = 0 THEN  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('삭제할 게시물이 없습니다. (ID: ' || pfreeBoardId || ')');  DBMS\_OUTPUT.PUT\_LINE('======================================================');  RETURN;  END IF;  -- 게시물 삭제  DELETE FROM tblFreeBoard  WHERE freeBoardId = pfreeBoardId;  COMMIT;  DBMS\_OUTPUT.PUT\_LINE('======================================================');  DBMS\_OUTPUT.PUT\_LINE('게시물이 삭제되었습니다. (게시물 ID: ' || pfreeBoardId || ')');  DBMS\_OUTPUT.PUT\_LINE('======================================================');  EXCEPTION  WHEN OTHERS THEN  ROLLBACK;  DBMS\_OUTPUT.PUT\_LINE('오류 발생: ' || SQLERRM);  END;  /  BEGIN  procDeleteFreeBoard('1');  END;  / | | | |