영역	문제	해설
DCL	정의	데이터 제어어 : 데이터베이스 사용자에게 권한을 부여하거나 회수하여 데이터 접근을 제어하는 명령어들을 포함. 데이터베이스의 보안 및 무결성을 유지하는 역할
DCL	SQL 명령어	- GRANT : 특정 사용자에게 권한을 부여 - REVOKE : 특정 사용자에게 부여된 권한 회수
DCL1	Commit과 Rollback의 장점	데이터 무결성 보장
DCL1	Commit과 Rollback의 장점	영구적인 변경을 하기 전에 데이터의 변경사항 확인 가능
DCL1	Commit과 Rollback의 장점	논리적으로 연관된 작업을 그룹핑하여 처리 가능
DDL	정의	데이터 정의어 : 데이터베이스의 구조, 즉 테이블, 뷰, 인덱스 등을 생성, 수정, 삭제하는 명령어들을 포함. 데이터베이스의 뼈대를 정의하는 역할 암시적(자동) COMMIT 기능, ROLLBACK 불가
DDL	SQL 명령어	- CREATE : 테이블, 뷰, 인덱스 등을 생성 - ALTER : 테이블 구조 변경 - DROP : 테이블, 뷰, 인덱스 등을 삭제 - TRUNCATE : 테이블의 모든 데이터를 삭제하고, 메모리 공간을 초기화(일반적으로 DELETE 보다 빠름.)
DDL1	ON DELETE CASCADE (CREATE, ALTER 제약 조건)	테이블 생성 시 ON DELETE CASCADE 옵션 지정 시, 참조하고 있던 부모테이블의 데이터가 삭제되면 이를 참조하는 자식 테이블의 데이터도 자동으로 함께 삭제됨.
DML	정의	데이터 조작어 : 데이터베이스에 저장된 데이터를 실제로 조작하는 명령어들을 포함. 데이터를 삽입, 조회, 수정, 삭제하는 작업을 수행 임시 저장 기능, COMMIT 전 ROLLBACK 가능
DML	SQL 명령어	- SELECT : 데이터를 조회 - INSERT : 데이터를 테이블에 삽입 - UPDATE : 데이터를 수정 - DELETE : 데이터를 삭제
DML1	DISTINCT 기능	DISTINCT 뒤에 * 를 사용할 수 있다.
DML1	DISTINCT 기능	DISTINCT 뒤에 나열되는 컬럼의 순서에 따라 결과의 순서만 달라지고, 집합의 수는 같다.
DML1	DISTINCT 기능	SELECT 문에서만 사용 가능하다.
DML1	DISTINCT 기능	DISTINCT 뒤에 나열되는 컬럼들의 중복값을 한 번만 출력하기 위해 사용한다.
DML2	쿼리문 실행순서1	FROM : *쿼리할 테이블을 지정, 여러 테이블이 있다면 이 단계에서 JOIN 처리 *** 쿼리 : 데이터베이스에 특정한 정보나 데이터를 요구하는 행위**
DML2	쿼리문 실행순서2	WHERE : FROM 절에서 가져온 데이터에서 특정 조건에 맞는 행들만 필터링
DML2	쿼리문 실행순서3	GROUP BY : WHERE 절을 통과한 행들을 특정 컬럼을 기준으로 그룹화
DML2	쿼리문 실행순서4	HAVING : GROUP BY 로 생성된 그룹들 중에서 특정 조건에 맞는 그룹만 필터링
DML2	쿼리문 실행순서5	SELECT : 최종적으로 반환할 컬럼이나 계산식을 결정(이때 컬럼의 별칭(alias)이 인식됨.)
DML2	쿼리문 실행순서6	ORDER BY : SELECT 절에서 선택된 최종 결과 집합을 지정된 순서로 정렬
DML3	NUMBER()	NUMBER(X, Y) 타입은 총 X자리 중 Y자리를 소수점 자리로 사용함. ex) NUMBER(5, 2) 일 때 100.00 가능 1000.00 불가, 100.000 삽입은 가능하나 입력되는 것은 100.00
DML4	CASE	DML 문의 일부로 사용되어 그 기능을 돕는 역할을 하는 표현식(도구), DML 명령어는 아님.(단독으로 실행될 수 있는게 아니기 때문에) 사용법 : CASE WHEN 대상컬럼, 조건 THEN 조건이 참 일 때 결과 값, ELSE 조건이 거짓일 때 결과 값, END ELSE 생략가능(생략 시 ELSE 조건일 경우 NULL 반환) ex) SELECT 학생이름, 성적, CASE WHEN 성적 > 90 THEN 'A' WHEN 성적 > 80 THEN 'B' ELSE 'F' END FROM 성적표;
DML5	DATE()	DATE() 타입 값 입력(INSERT) 방식 '- 표준 문자열 : DATE('YYYY'-'MM'-'DD') ex) DATE(2025-08-10) - 변환 함수 : TO_DATE('문자열', 'YYYY/MM/DD') ex) TO_DATE('20250813', 'YYYYMMDD') 또는 구분자 사용(어떠한 문자든 가능) TO_DATE('2025년08월13일', 'YYYY년MM월DD일') - 현재 날짜 함수 : SYSDATE 또는 SYSDATE +,- 연산자 ex) SYSDATE-10 = 현재 날짜 10일 전
DML6	INSERT 조건	컬럼 명시 후 값 지정 시 명시한 컬럼과 값 지정 개수가 일치해야 한다. ex) 이름, 나이, 성별 지정 시 이름, 나이, 성별 값 입력 컬럼을 명시하지 않을 경우 모든 컬럼 값 입력 필요(NULL 포함)
ER 모델링	정의	현실 세계의 정보를 데이터베이스에 저장하기 위해, 개체(Entity), 속성(Attribute), 관계(Relationship)를 이용하여 개념적 구조로 표현하는 방법 데이터베이스 설계의 초기 단계에서 사용되며, 시스템 내의 데이터 구조와 상호 작용을 명확하게 시각화 하는 데 도움을 준다.
ER 모델링	주요 구성 요소	- 개체(Entity) : 현실 세계에서 독립적으로 식별 가능한 유/무형의 객체나 개념을 의미 ex) 학생, 강의, 도서 - 속성(Attribute) : 개체의 특성이나 정보를 나타내는 항목 ex) 학생 개체는 학번, 이름, 학과 등의 속성을 가질 수 있음. - 관계(Relationship) : 개체들 간의 연관성이나 상호 작용을 의미 ex) 학생은 수강이라는 관계를 통해 강의와 연관될 수 있음.
ER 모델링	ER 모델의 장점	- 데이터 구조 이해 용이 : 데이터베이스의 전체적인 구조를 시각적으로 표현하여 데이터 관리 및 이해를 도움. - 설계 초기 단계 유용 : 데이터베이스 설계 초기 단계에서 데이터 요구사항을 명확히 정의하고 분석하는데 활용 - 정확성 향상 : ER 모델을 통해 데이터 간의 관계를 명확히 파악하여 데이터 무결성 및 일관성 유지에 기여

ER 모델링1	속성 설명	- 기본키 속성 : 엔터티를 식별할 수 있는 속성 - 외래키 속성 : 다른 엔터티의 관계에 포함되는 속성 - 일반 속성 : 다른 엔터티의 관계에 포함되지 않는 속성 - 복합 속성 : 여러 개의 속성이 모여 하나의 의미를 구성하는 속성 ex) 주소 : 도, 시, 동, 번지 등 여러 개의 하위 속성으로 구성 됨. - 파생 속성 : 다른 속성의 값으로부터 계산되어 얻어지는 속성 ex) 주문 총액 : 주문 항목들의 가격과 수량 합계
ER 모델링1	속성 설명2	- 엔터티에 대한 자세하고 구체적인 정보 - 업무상 인스턴스로 관리하고자 하는 더 이상 분리되지 않는 최소의 데이터 단위 - 정해진 주식별자에 함수적 종속성을 가져야 함. - 하나의 속성에 여러 개의 값이 있는 다중값일 경우 별도의 엔터티를 이용하여 분리 필요
ER 모델링2	관계(Relationship)와 Join	- 엔터티 간의 관계를 통해 데이터의 중복을 피하고, 각 데이터 요소를 한 번만 저장하여 유지관리의 복잡성을 줄일 수 있다. - Join 이란 식별자를 상속하고, 상속된 속성을 매핑키로 활용하여 데이터를 결합하는 것을 의미한다. - 부모의 식별자를 자식의 일반속성으로 상속하면 비식별 관계, 자식의 식별자에 포함하면 식별관계라고 할 수 있다.
ER 모델링2	관계의 구성 요소	- 선택성(Optionality) - 관계명 - 차수(Cardinality)
ER 모델링2	관계 설명	- 관계는 존재에 의한 관계와 행위에 의한 관계로 분류 '- 관계를 맺는다는 의미는 부모의 식별자를 자식에 상속하고, 상속된 속성을 매핑키(조인키)로 활용하는 것 '- 학생과 수업 간의 관계는 등록이라는 행위를 통해 형성되므로 행위 관계를 가진다. '- 주문항목은 반드시 주문이 있어야만 존재할 수 있으므로 존재 관계를 가진다.
ER 모델링2	관계 확인	엔터티 간의 관계는 주로 동사로 표현된다. 관계는 두 엔터티 사이에 어떤 행위나 상태가 일어나는지를 나타내기 때문 ex) 고객이 상품을 주문한다, 사원이 부서에 소속된다 같이 동사를 통해 관계를 파악하는 것이 일반적임. 관계를 찾기 위해 명사를 찾는 것은 부적절한 접근 엔터티 간의 관계 확인 시 올바른 접근법 예시 '- 관계연결에 대한 규칙이 서술되어 있는가? ex) 사원은 반드시 하나의 부서에 소속되어야 한다. - 두 개의 엔터티 사이에 정보의 조합이 발생되는가? ex) 고객 정보와 주문 정보를 조합하여 어떤 고객이 어떤 주문을 했는지 파악 - 두 개의 엔터티 사이에 관심 있는 연관규칙이 존재하는가? ex) 업무적인 관심사가 없다면 관계를 설정할 필요가 없음.
ER 모델링3	개체(Entity) 일반적인 특징	- 반드시 속성을 포함 - 업무 프로세스에 의해 이용되어야 함. - 유일한 식별자에 의해 식별이 가능해야 함. - 다른 엔터티와의 관계 형성 - 두 개 이상의 인스턴스를 가지는 집합, 사원, 부서, 상품 처럼 공통의 속성을 공유하는 개체들을 묶은 종류 또는 유형을 의미 ex) 사원 엔터티 : 사원번호, 이름, 직급, 소속부서 속성 보유
ER 모델링4	카디널리티 정의 및 계산식	카디널리티 : 특정 조건을 만족하여 반환될 것으로 예상되는 결과 집합의 행의 수 선택도 : 전체 데이터 중에서 특정 조건을 만족하는 데이터의 비율(0과 1사이의 값) 계산식 : 선택도 * 전체 레코드(전체 행의 개수) 수 ex) 전체 직원 레코드 수 : 1,000명인 테이블에서 직급 = 과장 조건의 카디널리티 조건 : 과장 직급 과장 직급의 선택도 : 0.1 (전체 직원의 10%가 과장이라고 가정) 카디널리티 : 0.1 * 1,000 = 100명
ER 모델링5	관계차수	엔터티 간의 관계에서 1:1, 1:M과 같이 관계의 기수성을 나타냄.
JOIN	FULL OUTER JOIN NATURAL JOIN	FULL OUTER JOIN : 양쪽 테이블의 모든 행을 결과에 포함시키며, 서로 일치하는 데이터가 없는 행은 비어있는 부분(컬럼)을 NULL로 채운다. NATURAL JOIN : 두 테이블에서 이름과 데이터 타입이 같은 컬럼을 모두 찾아서, 그 컬럼들을 기준으로 자동으로 등가 조인(Equi-join)을 수행하는 방식 NATURAL JOIN의 위험성 '- 우연히 이름이 같은 컬럼이 있다면 의도치 않게 해당 컬럼까지 모두 조인 조건에 포함되어 잘못된 결과를 낼 수 있음. - 조건이 코드에 명시적으로 드러나지 않아, 다른 개발자가 코드를 이해하고 유지보수하기 어려움.
NULL	NULL 함수	NULLIF(COL A, COL B) : A와 B가 동일하면 NULL을, 동일하지 않으면 A를 반환
NULL	NULL 함수	NVL(표현식, 대체값) : 표현식이 NULL인 경우 대체 값을 반환, 그렇지 않으면 표현식 반환
NULL	NULL 함수	COALESCE(표현식1, 표현식2, ... 표현식N) : 주어진 표현식들을 순서대로 평가하여 NULL이 아닌 값이 나오면 즉시 해당 값 반환, 모든 표현식이 NULL 일 경우 NULL 반환
NULL	NULL 정렬	- ORDER BY COMM DESC NULLS LAST 시 NULL이 맨 뒤에 배치 - ORDER BY COMM DESC 시 NULL이 맨 앞에 배치 - ORDER BY COMM NULLS FIRST 시 NULL이 맨 앞에 배치 - ORDER BY COMM 시 NULL이 마지막에 배치
VIEW	정의	데이터베이스 내의 가상 테이블, 하나 이상의 테이블이나 다른 뷰를 기반으로 생성되며, 실제 데이터를 저장하지 않고, 쿼리 결과를 논리적으로 테이블 형태로 보여준다.
VIEW	특징	- 가상 테이블 : 물리적인 저장 공간을 차지하지 않으며, 사용할 때마다 뷰 정의에 따라 쿼리가 실행되어 결과를 반환 - 쿼리 단순화 : 복잡한 쿼리를 뷰로 만들어서 사용하면, 쿼리 작성 및 실행을 간소화 할 수 있음. - 보안 강화 : 특정 사용자에게 필요한 데이터 열만 접근 가능하도록 제한 가능 - 데이터 독립성 : 기반 테이블의 구조 변경으로부터 영향을 덜 받음, 뷰를 수정하지 않고도 기반 테이블을 변경할 수 있으며, 사용자는 뷰를 통해 일관된 인터페이스를 유지 가능 - 논리적 모델링 : 데이터베이스의 논리적 구조를 표현하는 데 유용, 복잡한 데이터 관계를 단순화
VIEW	주의사항	- 삽입, 수정, 삭제 연산이 제한 될 수 있음, 기반 테이블의 구조에 따라 가능할 수도 있고, 불가능할 수도 있음. - 복잡한 뷰는 성능 저하를 유발할 수 있음.
그룹 함수	그룹 함수 종류	GROUP BY : 지정된 하나 이상의 열을 기준으로 행들을 그룹화하고, 각 그룹에 대해 집계 함수(SUM, COUNT, AVG 등)를 적용하여 요약된 결과를 반환 주요 용도 : 단일 레벨의 데이터 요약 ex) 부서별 평균 급여 계산하기 **select** department_id, avg(salary) **from** employees **group by** department_id; 결과 : 부서별 평균 급여 + group by 절에 포함되지 않은 컬럼으로 정렬할 수 없다.

그룹 함수	그룹 함수 종류	ROLLUP : GROUP BY의 확장 기능으로, 지정된 열들의 계층 구조를 기반으로 소계(SUB TOTAL)와 총계(GRAND TOTAL)를 자동으로 생성 ROLLUP에 명시된 열의 순서가 매우 중요하며, 오른쪽에서 왼쪽으로 하나씩 그룹핑을 해제하며 소계를 계산 주요 용도 : 계층적인 보고서, 월별/분기별/연간 실적 ex) 지역별, 부서별 인원수 및 지역별 소계, 전체 총계 계산하기 **select** region, department, count(*) **from** sales **group by rollup**(region, department); 결과 : 각 지역의 각 부서별 인원수, 각 지역별 소계(모든 부서의 인원수 합계), 전체 총계(모든 지역, 모든 부서의 총 인원수)
그룹 함수	그룹 함수 종류	CUBE : GROUP BY의 확장기능, ROLLUP 보다 더 포괄적임. 지정된 열들로 만들 수 있는 모든 가능한 조합에 대한 소계와 총계를 생성 명시된 열들의 모든 조합에 대한 집계 결과 제공 주요 용도 : 다차원적인 데이터 분석, 교차 분석표 ex) 지역별, 제품별 판매량 및 각 항목별 소계, 전체 총계 계산하기 **select** region, rpoduct, sum(sales_amount) **from** sales **group by cube**(region, product); 결과 : 각 지역의 각 제품별 판매량, 각 지역별 소계(모든 제품의 판매량 합계), 각 제품별 소계(모든 지역에서의 판매량 합계), 전체 총계
그룹 함수	그룹 함수 종류	GROUPING SETS : 사용자가 원하는 그룹 조합만을 명시적으로 지정하여 결과를 얻을 수 있게 해주는 가장 유연한 그룹 함수 주요 용도 : 복잡하고 특정 조건의 그룹 요약이 필요할 때 ex) 부서별-직급별 인원수와 부서별 총 인원수만 계산하기 (직급별 총 인원수는 제외) **select** department, job_title, count(*) **from** employees **group by grouping sets**((department, job_title), (department)); 결과 : (부서, 직급) 조합의 인원수와 (부서) 조합의 인원수
데이터베이스 개념	식별자 종류(근원 구분)	- 본질 식별자 : 비즈니스(현실)에서 의미가 있는 값, 변경 가능성, 업무적 의미 보유 ex) 주민등록번호, 이메일... - 인조 식별자(내부식별자) : 시스템이 인공적으로 부여한 값, 불변성, 업무적 의미 없음. ex) 자동 생성 ID, 회원번호...
데이터베이스 개념	식별자 종류(구조 구분)	- 단일 식별자 : 하나의 컬럼으로 구성, 구조 단순 ex) ID... - 복합 식별자 : 둘 이상의 컬럼으로 구성, 두 값의 조합이 유일성을 보장 ex) 수강신청 내역에서 식별자는 학번과 과목코드 조합
데이터베이스 설계	데이터베이스의 3단계(3층) 스키마 구조 중 1단계	- 외부 스키마 : 특정 사용자나 어플리케이션이 접근하는 데이터베이스의 부분을 정의 사용자 개인, 응용 프로그램 관점 하나의 데이터베이스에는 여러 개의 외부 스키마가 존재할 수 있음. ex) 영업팀용 스키마, 인사팀용 스키마 사용자는 외부 스키마를 통해 데이터베이스에 접근하므로, 내부 구조나 전체 논리 구조를 알 필요가 없음. 데이터 단순화, 보안 수준 향상 논리적 독립성 제공 DML(SELECT, INSERT 등)
데이터베이스 설계	데이터베이스의 3단계(3층) 스키마 구조 중 2단계	- 논리 스키마(개념 스키마) : 데이터베이스의 전체적인 논리적 구조와 제약조건을 정의 기관 전체, 통합적 관점(개발자, 데이터베이스 관리자의 관점) 데이터베이스에 무엇을 저장할지를 정의(모든 개체와 그 속성, 개체들 간 관계, 보안 규칙 및 무결성 제약 조건) 모든 외부 스키마는 논리 스키마를 기반으로 생성 DDL(CREATE, ALTER 등)
데이터베이스 설계	데이터베이스의 3단계(3층) 스키마 구조 중 3단계	- 내부 스키마 : 데이터가 물리 저장 장치(하드디스크 등)에 실제로 어떻게 저장되는지를 정의 물리적 저장 장치 관점(시스템 프로그래머, 하드웨어 설계자의 관점) 데이터베이스의 성능 및 효율성과 직접적인 관련이 있음. 논리 스키마나 외부 스키마에 영향을 주지 않으면서 저장 구조를 변경 가능 물리적 독립성 제공
데이터베이스 설계	객체명 규칙 (테이블명, 컬럼명 등)	- 반드시 문자로 시작해야 함.(숫자로 시작할 수 없음.) - 이름에는 문자, 숫자, 그리고 일부 특수문자(주로_)를 사용할 수 있음. - 하이픈(-)과 같은 다른 특수문자는 연산자(빼기)로 인식될 수 있어 사용 지양 - SQL 예약어(ex SELECT, TABLE 등)는 사용 지양
정규화	정의	정규화는 데이터의 중복을 최소화하고 데이터 무결성을 보장하기 위해, 관계형 데이터베이스의 테이블 구조를 체계적으로 분해하는 과정 잘못된 설계로 인해 발생할 수 있는 문제들을 사전에 방지하기 위해 테이블을 잘 나누는 작업
정규화	사용목적	정규화 되지 않은 테이블에서는 데이터를 추가, 수정, 삭제할 때 이상 현상(Anomaly)이 발생할 수 있음. '- 삽입 이상 : 불필요한 데이터 없이는 원하는 데이터를 삽입할 수 없는 문제 '- 갱신 이상 : 중복된 데이터 중 일부만 수정되어 데이터의 일관성이 깨지는 문제 '- 삭제 이상 : 특정 데이터를 삭제하면, 유지해야 할 다른 정보까지 함께 삭제되는 문제 정규화는 이러한 이상 현상을 방지하고, 데이터베이스를 더 효율적이고 논리적으로 관리할 수 있게 함.
정규화	정규화 단계 설명	정규화는 여러 단계로 이루어져 있으며, 일반적으로 3차 정규형(3NF)까지 만족시키는 것을 목표로 함. 정규화는 하나의 테이블은 하나의 주제만 다루도록 테이블을 잘 분해하여 데이터의 중복과 각종 이상 현상을 해결하는 과정
정규화	제1정규형(1NF)	테이블의 모든 컬럼(속성) 값은 원자 값 이어야 함. 한 칸의 셀에는 오직 하나의 값만 들어가야 한다는 의미
정규화	제2정규형(2NF)	테이블이 제1정규형을 만족하고, 기본 키(Primary Key)의 일부에만 종속되는 컬럼(부분 함수 종속)이 없어야 함. 기본 키가 복합 키로 구성된 경우에 의미가 있음.
정규화	제2정규형(2NF) 예시	우측 테이블의 기본 키는 학생 ID, 과목명임. 하지만 지도교수는 과목명에만 종속되므로 부분 함수 종속 발생 (문제점 : 학생 1001이 데이터베이스 수강을 취소하면 이교수 라는 정보까지 사라짐.) -> 해결 : 과목별 지도교수 별도 테이블 분리
정규화	제3정규형(3NF)	테이블이 제2정규형을 만족하고, 기본 키가 아닌 다른 컬럼에 종속되는 컬럼(이행 함수 종속)이 없어야 함.
정규화	제3정규형(3NF) 예시	우측 테이블의 기본 키는 학생 ID, 학과장은 학과에 의해 결정되고, 학과는 학생 ID에 의해 결정됨. 학생 ID -> 학과 -> 학과장 이행종속 발생 (문제점 : 학생 501이 자퇴하면 컴퓨터 공학 학과장이 김박사 라는 정보까지 사라짐.) -> 해결 : 학과별 학과장 테이블 별도 분리

제2정규형(2NF) 위반 테이블 예시

학생 ID	과목명	성적	지도교수
1001	데이터베이스	A	이교수
1001	운영체제	B	박교수

SP	PL/SQL 설명	- PL/SQL(Procedural Language/SQL)은 오라클 데이터베이스에서 사용하는 SQL의 확장 언어 - PL/SQL에서 name이라는 변수에 'aaa'를 대입할 경우 ":="을 사용 - PL/SQL은 절차형 언어로 PL/SQL 내부에서 테이블을 생성할 수 있다. - PL/SQL에서 조건문은 IF~ THEN~ ELSEIF~ END IF 와 CASE~WHEN을 사용한다. - PL/SQL문의 기본 구조로 DECLARE, BEGIN ~ END는 필수적으로 사용해야 한다. EXCEPTION은 선택사항 - Procedure 내부에 작성된 절차적 코드는 PL/SQL 엔진이 처리하고 일반적인 SQL 문장은 SQL 실행기가 처리한다. - Procedure, User Defined Function, Trigger 객체를 PL/SQL로 작성할 수 있다. - 변수와 상수 등을 사용하여 일반 SQL문장을 실행할 때 WHERE 절의 조건 등으로 대입할 수 있다.
계층형 질의	계층형 질의1	CONNECT_BY_ISLEAF (의사 컬럼) : 해당 행이 계층 구조의 가장 마지막 노드(리프 노드, 자식이 없는 노드)인지를 확인 후 리프 노드이면 1, 아니면(자식이 있으면) 0을 반환
계층형 질의	계층형 질의2	CONNECT_BY_ROOT : 현재 행의 최상위 루트 노드의 특정 컬럼 값을 반환
계층형 질의	계층형 질의3	SYS_CONNECT_BY_PATH(컬럼, 구분자) : 루트 노드부터 현재 노드까지의 경로를 문자열로 표시
계층형 질의	계층형 질의4	ORDER SIBLINGS BY : 계층 구조는 유지하면서, 같은 레벨의 형제 노드들 사이의 정렬 순서를 지정
계층형 질의	계층형 질의5	CONNECT_BY_ISCYCLE : 계층형 쿼리에서 데이터의 무한 루프 발생 여부를 확인, 현재 행을 자식으로 포함시킬 때 무한 루프가 생긴다면 1을, 그렇지 않다면 0을 반환
계층형 질의	계층형 질의6	NOCYCLE : 계층형 쿼리 실행 중 무한 루프(순환 구조)를 발견하더라도 오류를 발생시키지 않고 계속 진행하도록 만드는 옵션
단일행 함수	단일행 함수1	ADD_MONTHS : 지정한 날짜에서 n개월 이후 날짜를 출력하는 함수 NEXT_DAY(날짜,요일)는 지정한 날짜 뒤 의 첫 번째 지정요일에 해당하는 날짜를 반환
데이터 무결성	테이블에 대한 설명	- 하나의 테이블은 반드시 한 계정의 소유여야 한다. - 소유자가 다른 경우 테이블명은 같은 이름으로 생성 가능하다. - 하나의 행의 하나의 컬럼에는 반드시 하나의 값만 삽입되어야 한다. - 테이블 생성 시 정의한 컬럼의 데이터 유형은 테이블 생성 이후 ALTER 명령어로 변경 가능하다.
데이터베이스 개념	도메인	- 도메인은 속성에 대한 값의 범위 등 제약사항을 기술할 수 있다. - 특정 속성이 가질 수 있는 값의 범위나 집합을 의미(해당 컬럼에 입력될 수 있는 데이터의 자격 조건 또는 규칙) - 데이터의 무결성과 일관성을 보장하는 역할 - 테이블의 각 컬럼이 의미적으로 올바른 데이터만을 갖도록 강제하는 논리적인 제약 조건 ex) 컬럼 : 성별 도메인 : 남, 여 컬럼 : 나이 도메인 : 0 이상 150 이하의 정수 컬럼 : 학점 도메인 : A+, A, B+, B, C+, C, D, F
데이터베이스 개념	데이터 모델링 사용 목적	- 데이터베이스를 구축하기 위한 용도와 업무에 대한 분석 및 표현 - 데이터모델링 자체로서 업무의 흐름을 설명하고 분석하는 부분에 의미를 가지고 있다.(데이터 모델은 업무 규칙과 데이터 간의 관계를 시각적으로 표현하므로, 모델 자체만으로도 업무를 이해하고 분석) - 분석된 모델을 가지고 데이터베이스를 생성하여 개발 및 데이터관리에 사용하기 위한 것이다.(데이터 모델링의 실질적인 목표 : 데이터베이스 스키마를 생성하는 직접적인 설계도 역할) - 업무정보를 구성하는 기초가 되는 정보들에 대해 일정한 표기법에 의해 표현한다.(데이터 모델링의 정의 : 추상적인 업무 정보를 ERD와 같은 표준화된 표기법을 통해 명확하고 논리적으로 표현하는 과정)
데이터베이스 개념	관계형 데이터베이스 특징	- 테이블 기반 구조(테이블 기반 데이터 저장) - 관계를 통해 데이터 연결 - 선언형 언어(SQL) 사용 - 선언적(Declarative) 방식 : 처리 과정이나 순서는 명시하지 않고, 단순히 원하는 결과가 무엇인지만을 선언, 어떻게 가져올지는 데이터베이스 관리 시스템(DBMS)이 알아서 최적의 방법을 찾아 처리 - 절차적 방식과 반대 개념
서브쿼리	스칼라 서브쿼리	- 쿼리의 결과로 단 하나의 값(하나의 행, 하나의 열) 만을 반환, 마치 하나의 상수나 변수처럼 취급할 수 있음. - 사용위치 : SELECT 절, WHERE 절, HAVING 절, ORDER BY 절 등 단일 값이 올 수 있는 거의 모든 곳 - 데이터 타입 호환성 : 외부 쿼리의 컬럼과 데이터 타입이 호환되어야 함. - 메인쿼리와 스칼라 서브쿼리의 연결 조건이 필요하다면 반드시 스칼라 서브쿼리에 정의해야 함. - 하나의 행에 해당하는 스칼라 서브쿼리 결과가 0건이더라도 메인쿼리절에서 생략을 하지 않는 한 NULL로 출력
서브쿼리	단일행 서브쿼리	- 쿼리의 결과로 단 하나의 행과 하나 이상의 열을 반환(하나 이상의 열을 반환하는 점에서 스칼라 서브쿼리와 차이가 있음.)
서브쿼리	서브쿼리 설명	- 다중행 연산자는 IN, ANY, ALL이 있으며 서브쿼리의 결과로 하나 이상의 데이터가 RETURN 됨. - INLINE VIEW는 FROM절에 사용되는 서브쿼리로서 실질적인 OBJECT는 아니지만 SQL 문장에서 마치 VIEW나 테이블 처럼 사용함. - TOP-N 서브쿼리는 INLINE VIEW의 정렬된 데이터를 ROWNUM을 이용해 결과 행 수를 제한하거나 TOP(N) 조건을 사용함. - 상호 연관 서브쿼리는 서브쿼리가 메인쿼리의 행 수 만큼 실행되는 쿼리로서 실행 속도가 상대적으로 떨어지는 SQL 문장임. 그러나 복잡한 일반 프로그램을 대체할 수 있기 때문에 조건에 맞는다면 적극적인 검토가 필요 - 연관 서브쿼리 : 서브쿼리가 메인쿼리 컬럼을 가지고 있는 것, 주로 메인쿼리가 먼저 수행된 후에 서브쿼리에서 조건이 맞는지 확인
윈도우 함수	순위 함수	- RANK() : 동일한 값에는 같은 순위를 부여하고, 다음 순위는 중간 순위를 건너뛰어 계산(일반적인 공동순위) - DENSE_RANK() : 동일한 값에는 같은 순위를 부여하지만, 다음 순위를 건너뛰지 않고 순차적으로 순위 부여 - ROW_NUMBER() : 동일한 값에 상관없이 고유한 순위를 순차적으로 부여(같은 값일 경우 데이터 순서에 따라 순위 부여) - NTILE(n) : 전체 데이터를 지정된 개수(n)만큼의 그룹으로 나누고, 각 행이 어느 그룹에 속하는지 번호를 부여 ex) NTILE(4) : 다섯 개의 행이 있을 경우 1,2행 1그룹 나머지 2,3,4행 2,3,4그룹
윈도우 함수	윈도우 함수 설명	- GROUP BY는 실제 출력되는 행을 줄여서 출력하나, 윈도우 함수는 실제 행이 줄어들지 않는다. 그러므로, 병행작성이 불가능 한 것은 아니지만, 병행하여 사용하지 않고 필요에 따라 둘 중 하나를 선택해 사용한다. - SUM, MIN, MAX 등과 같은 집계 윈도우 함수를 사용할 때 WINDOW 절과 함께 사용하면 집계의 대상이 되는 레코드 범위를 지정할 수 있다.

제3정규형(3NF) 위반 테이블 예시

1002	데이터베이스	A	이교수

학생ID	학과	학과장
501	컴퓨터공학	김박사
502	통계학	최박사
503	컴퓨터공학	김박사

윈도우 함수	윈도우 함수와 윈도우 절	- 윈도우 함수 : 테이블의 특정 행 그룹(이를 윈도우 라고 부름)을 대상으로 값을 계산하는 함수, 일반 집계 함수처럼 결과를 한 줄로 요약하는 대신, 각 행마다 계산된 값을 반환 -- 윈도우 함수의 종류 --- 순위 함수 --- 집계 함수(SUM, AVG, COUNT, MAX, MIN) --- 행 순서 함수 ---- LAG(컬럼, n) : 현재 행보다 n번째 이전 행의 값을 가져온다. ---- LEAD(컬럼, n) : 현재 행보다 n번째 다음 행의 값을 가져온다. ---- FIRST_VALUE(컬럼) : 파티션 내의 첫 번째 행의 값을 가져온다. ---- LAST_VALUE(컬럼) : 파티션 내의 마지막 번째 행의 값을 가져온다. - 윈도우 절 : 그 계산 범위를 현재 행 기준으로 더 정밀하게 지정하는 옵션 -- 윈도우 절의 종류 --- PARTITION BY 절 : 계산 대상을 그룹(파티션)으로 나눈다. GROUP BY와 비슷하지만, 행들을 하나로 합치지 않고 각 행을 그대로 유지하며 계산을 수행 --- ORDER BY 절 : 파티션 내에서 행의 순서를 정한다. 순위 함수나 누적 계산을 할 때 어떤 순서로 값을 처리할지 지정하는 필수 요소 --- 프레임 절(ROWS/RANGE) : ORDER BY 까지 지정된 파티션 내에서, 현재 행을 기준으로 계산에 포함할 움직이는 범위를 더 세밀하게 지정 ---- ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW : 파티션의 처음부터 현재 행 까지 (누적 계산에 사용) ---- ROWS BETWEEN n PRECEDING AND CURRENT ROW : n번째 이전 행부터 현재 행 까지(이동 평균에 사용) ---- ROWS BETWEEN CURRENT ROW AND UNBOUNDED FOLLOWING : 현재 행부터 파티션의 끝까지