Day 1

1장 데이터베이스 시작하기

안녕하세요! <코딩 자율학습 SQL 데이터베이스 입문>을 시작하는 여러분 반갑습니다. 이 책을 공부하는 데 조금 더 도움을 드리고자 학습 가이드를 준비했습니다. 학습 가이드에서는 장별로 배운 내용을 다시 한번 정리하고 문제를 풀어봅니다. 이를 통해 SQL 문법을 완전히 내 것으로 만들고 다양하게 응용하는 능력을 기를 수 있을 것입니다.

첫 날에는 데이터베이스, DBMS, SQL이란 무엇인지 알아보고 MySQL 실습 환경을 설정합니다. 오늘 공부할 내용은 다음과 같습니다.

🧐 공부할 내용(19~54p)

- 데이터베이스란
- 데이터 저장 형식
- MySQL 실습 환경 설정하기

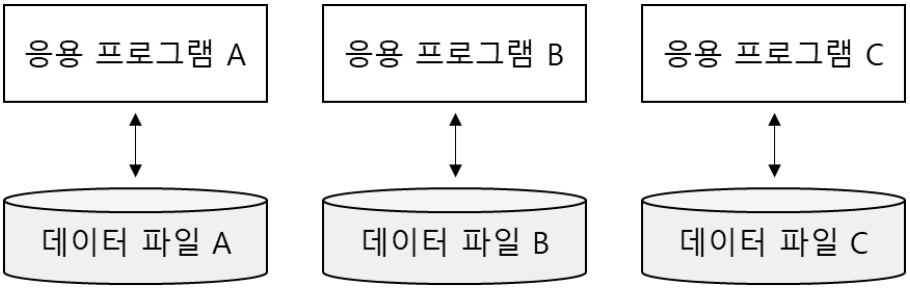
책에서 기본적인 내용을 익히므로 여기서는 데이터베이스, DBMS, SQL이 어떤 배경을 가지고 등장했는지 알아보겠습니다.

1. 데이터베이스와 DBMS의 등장 배경

데이터베이스(DataBase, DB)는 데이터를 좀 더 효율적이고 안전하게 저장, 관리, 활용하기 위해 고안됐습니다. 정보 기술의 발전과 함께 데이터의 양이 급격히 증가하고 데이터를 더 잘 조직화할 방법이 필요해지면서 발전하게 됐습니다.

데이터베이스가 등장하기 전에는 데이터를 파일 시스템을 통해 관리했습니다. 파일 시스템(file system)은 데이터를 파일 형태로 저장하고 프로그램이 직접 파일에서 데이터를 읽고 쓰는 방식입니다. 일반적으로 활용하는 메모장 파일, 엑셀 파일 등을 생각하면 됩니다.

▼ 파일 시스템의 개념



그런데 파일 시스템에는 문제가 있었습니다.

파일 시스템의 문제

첫째, 데이터 중복으로 인한 데이터 불일치 문제입니다. 파일 시스템에서 동일한 데이터가 여러 파일에 저장될 경우 데이터가 서로 불일치하게 되는 문제가 발생할 수 있습니다. 예를 들어 고객 정보가 여러 파일에 저장됐는데 고객 전화번호가 바뀌어 한 파일에서만 수정하고 다른 파일에서 수정하지 않으면 데이터 불일치 문제가 발생합니다.

둘째, 데이터 접근의 비효율 문제입니다. 파일 시스템에서 데이터를 검색하거나 수정하면 시간이 오래 걸리고 고급 검색 기능도 부족합니다.

셋째, 보안 문제입니다. 파일 시스템에선 파일 수준의 접근 제어만 가능해 개별 데이터에 대한 세부적인 접근 권한 설정이 어렵습니다.

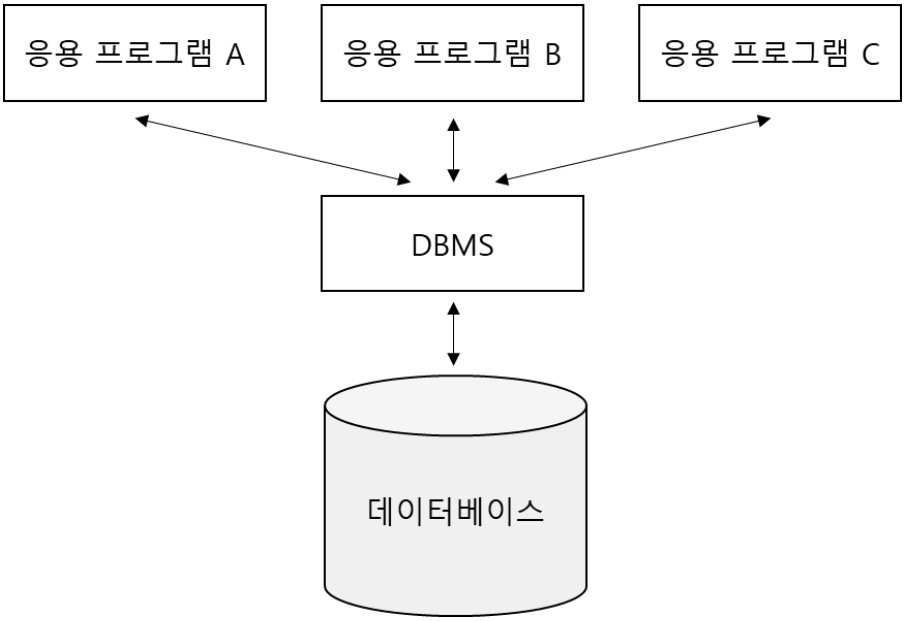
넷째, 데이터 무결성 유지의 어려움입니다. 파일 시스템에 데이터를 저장할 때 데이터가 특정 규칙을 준수하도록 강제하는 방법이 거의 없습니다. 예를 들어 파일 시스템에 고객 ID를 저장할 때 고객 ID는 고유해야 한다는 규칙을 강제할 수 없습니다.

데이터베이스와 **DBMS**의 등장

파일 시스템의 문제를 해결하기 위해 데이터베이스 개념이 도입됐습니다. 데이터베이스는 데이터를 체계적으로 관리하고, 중복을 줄이며, 일관성을 유지할 수 있는 구조화된 공통의 저장소를 제공합니다.

데이터베이스는 데이터를 저장, 관리하기 위한 개념이기 때문에 이를 사용하려면 전문 소프트웨어가 필요합니다. 이에 데이터베이스를 생성하고 관리할 수 있는 도구로 **DBMS**(Database Management System)가 등장했습니다.

▼ **DBMS**의 개념



DBMS의 기능은 책에서도 설명했지만 특히 기억할 내용을 꼽으라면 다음의 3가지입니다.

첫째, DBMS는 데이터를 효율적으로 관리해 줍니다. DBMS는 데이터를 조직화, 색인화, 최적화해 제공하므로 원하는 데이터에 빠르게 접근할 수 있습니다.

둘째, DBMS는 데이터 보안 및 무결성을 유지시켜 줍니다. DBMS는 데이터를 중앙에서 관리하므로 데이터별로 권한 설정, 무결성 제약 조건 등을 적용할 수 있습니다.

셋째, DBMS는 다중 사용자가 데이터베이스를 동시에 이용할 수 있도록 제어합니다. 여러 사용자가 동시에 데이터에 접근하더라도 충돌 없이 관리해 줍니다.

2. SQL의 등장 배경

데이터베이스는 단순한 데이터 저장소 역할뿐만 아니라 데이터 간의 관계를 정의하고 데이터 수정 시에도 관계를 유지할 수 있도록 하는 '데이터 모델'을 제공합니다.

데이터 모델의 종류로는 데이터를 부모-자식 관계로 저장하는 '계층형 데이터 모델', 데이터 간의 관계를 그래프 구조로 저장하는 '망형 데이터 모델', 데이터를 테이블 형태로 저장하는 '관계형 데이터 모델' 등이 있습니다. 그중에서 널리 사용되는 데이터 모델은 관계형 데이터 모델이며, SQL은 바로 관계형 데이터 모델을 조작하기 위한 쿼리 언어입니다.

1970년대 후반 관계형 모델 기반의 관계형 데이터베이스(Relational Database)와 이를 조작하는 언어인 SQL(Structured Query Language)이 등장하면서 데이터베이스 기술은 급격히 발전했습니다.

관계형 데이터베이스는 데이터를 행(Row)과 열(Column)로 구성된 테이블 형태로 저장하고 테이블 간의 관계를 정의해 복잡한 데이터 구조를 간단히 표현합니다. 또한 SQL을 사용해 데이터를 효율적으로 조회, 수정, 삭제합니다.

관계형 데이터베이스를 관리하는 DBMS로는 이 책에서 실습하는 MySQL 외에도 PostgreSQL, Oracle, SQL Server 등이 있습니다. 이들은 모두 SQL을 통해 데이터를 조작, 관리합니다.

🥕학습 Tip

관계형 데이터베이스를 처음 학습할 때는 기본 개념 이해와 SQL 실습을 병행하며 반복 학습을 통해 점진적으로 공부하는 것이 좋습니다. SQL 문법을 일부러 외우지 않아도 몸으로 익힐 수 있을 것입니다.

🥕

Day 2

2장 데이터 생성·조회·수정·삭제하기

안녕하세요! 오늘은 2장 <데이터 생성·조회·수정·삭제하기>를 살펴봅니다.

🧐 공부할 내용(55~86p)

- 데이터 CRUD란
- 데이터베이스 만들기
- 데이터 삽입 및 조회하기
- 데이터 수정 및 삭제하기

1. 테이블 만들기

테이블을 만들 때는 **CREATE TABLE** 문을 사용합니다.

```
CREATE TABLE 테이블명 (
  칼럼명1 자료형1,
  칼럼명2 자료형2,
  ...
  PRIMARY KEY (칼럼명)
);
```

직원 정보를 저장하는 employees 테이블은 다음과 같이 생성합니다.

```
CREATE TABLE employees (
  id INTEGER,            -- 직원 ID
  name VARCHAR(50),      -- 이름
  position VARCHAR(50),  -- 직책
  salary DECIMAL(10, 2), -- 급여
  hire_date DATE,        -- 입사일
  PRIMARY KEK (id)       -- 기본키 지정: id
);
```

테이블을 만들 때는 적절한 테이블명과 컬럼명을 사용해야 합니다. 이름만 봐도 의미가 명확하고 데이터의 목적을 잘 나타내야 합니다.

2. 데이터 삽입

테이블에 데이터를 삽입할 때는 **INSERT INTO** 문을 사용합니다.

```
INSERT INTO 테이블명 (칼럼명1, 칼럼명2, ...)
VALUES (입력값1, 입력값2, ...);
```

employees 테이블에 직원 데이터를 삽입하면 다음과 같습니다.

```
INSERT INTO employees (id, name, position, salary, hire_date)
VALUES (1, '김철수', '개발자', 5000000, '2022-03-01');
```

테이블에 데이터를 삽입할 때는 칼럼에 정의된 자료형에 맞게 데이터를 삽입해야 합니다. 숫자형(INTEGER)으로 정의된 칼럼에 문자열을 삽입하거나 문자열(VARCHAR)로 정의된 칼럼에 숫자를 삽입하면 안 됩니다. 또한 INSERT INTO 문에서 지정한 칼럼의 순서와 VALUES의 데이터 순서가 반드시 일치해야 합니다.

3. 데이터 조회

테이블에 데이터를 조회할 때는 **SELECT** 문을 사용합니다.

```
SELECT 칼럼명1, 칼럼명2, ...
FROM 테이블명
WHERE 조건;
```

직책이 '개발자'인 직원의 이름, 직책, 급여를 조회하면 다음과 같습니다.

```
SELECT name, position, salary
FROM employees
WHERE position = '개발자';
```

SELECT 문은 데이터를 조회할 때 사용하는 SQL의 가장 기본적인 명령어입니다. 하지만 잘못 사용하면 성능 저하를 초래할 수 있습니다. 예컨대 SELECT * 절로 테이블의 모든 칼럼을 조회하거나 WHERE 절에 조건 없이 대량의 데이터를 조회하면 비효율적입니다. 따라서 SELECT 절에는 필요한 칼럼만 명시적으로 지정하고, WHERE 절에 조건을 지정해 데이터만 필터링해 조회하는 것이 좋습니다.

4. 데이터 수정

테이블에 데이터를 수정할 때는 **UPDATE** 문을 사용합니다.

```
UPDATE 테이블명
SET 칼럼명 = 입력값
WHERE 조건;
```

id가 1번인 직원의 급여를 6,000,000원으로 수정하면 다음과 같습니다.

```
UPDATE employees
SET salary = 6000000
WHERE id = 1;
```

UPDATE 문을 잘못 사용하면 데이터 손실이나 시스템 오류를 초래할 수 있습니다. 데이터를 안전하고 효율적으로 수정하기 위해 WHERE 절로 수정 대상을 명확히 지정하고, 중요한 데이터를 수정할 때는 백업을 해두는 것이 좋습니다. 또한 한꺼번에 많은 튜플을 수정할 때 버벅일 수 있으므로 100개씩 수정하는 식으로 범위를 나눠 작업하는 것도 방법입니다.

5. 데이터 삭제

테이블의 데이터를 삭제할 때는 **DELETE** 문을 사용합니다. DELETE 문은 튜플 단위로 데이터를 삭제합니다.

```
DELETE FROM 테이블명
WHERE 조건;
```

id가 1번인 직원 데이터는 다음과 같이 삭제합니다.

```
DELETE FROM employees
WHERE id = 1;
```

DELETE 문 사용 시 주의할 점은 UPDATE 문과 비슷합니다. 데이터를 안전하고 효율적으로 삭제하기 위해 WHERE 절로 삭제 대상을 명확히 지정하고, 중요한 데이터라면 삭제하기 전 백업을 해두는 것이 좋습니다. 또한 한 테이블에만 있는 데이터가 아니라 여러 테이블에 있는 데이터라면 연쇄적으로 삭제해야 무결성을 유지할 수 있습니다. 따라서 삭제 전에 데이터 관계를 점검해 삭제해도 되는지 안 되는지 판단합니다.

Day 3

2장 데이터 생성·조회·수정·삭제하기

오늘은 2장 <데이터 생성·조회·수정·삭제하기>를 연습합니다.

🧐 공부할 내용(87p)

- 셀프체크

셀프체크를 모두 잘 풀었나요? 그렇다면 추가 연습 문제를 풀면서 실력을 키워 봅시다.

1. 데이터 CRUD 연습 문제

문제 1

다음 데이터를 저장하는 employees 테이블을 만드세요. id(ID), salary(급여) 칼럼은 정수형, name(직원명), department(부서), position(직책)은 최대 50자를 저장하는 문자형으로 지정합니다.

employess

id	name	department	position	salary
1	김철수	개발	사원	3500000
2	박영희	개발	대리	4200000
3	이민호	기획	팀장	5500000
4	최수진	기획	사원	3300000
5	정하늘	영업	사원	3100000
6	오준수	영업	대리	4000000
7	서지우	마케팅	팀장	6000000
8	이은지	마케팅	사원	3200000
9	안현준	개발	팀장	5800000
10	홍길동	영업	사원	3000000

정답:

```sql
-- employees 테이블 생성
CREATE TABLE employees (
  id INTEGER,              -- ID
  name VARCHAR(50),        -- 직원명
  department VARCHAR(50), -- 부서
  position VARCHAR(50),    -- 직책
  salary INTEGER,          -- 급여
  PRIMARY KEY (id)
);
```

문제 **2**

employees 테이블에 모든 직원 정보를 삽입하세요.

정답:

```sql
-- employees 데이터 삽입
INSERT INTO employees (id, name, department, position, salary)
VALUES
(1, '김철수', '개발', '사원', 3500000),
(2, '박영희', '개발', '대리', 4200000),
(3, '이민호', '기획', '팀장', 5500000),
(4, '최수진', '기획', '사원', 3300000),
(5, '정하늘', '영업', '사원', 3100000),
(6, '오준수', '영업', '대리', 4000000),
(7, '서지우', '마케팅', '팀장', 6000000),
(8, '이은지', '마케팅', '사원', 3200000),
(9, '안현준', '개발', '팀장', 5800000),
(10, '홍길동', '영업', '사원', 3000000);
```

문제 **3**

새로운 직원 '장미희'를 '기획' 부서의 '사원'으로 추가하세요. 급여는 3,400,000원입니다.

정답:

```sql
INSERT INTO employees (id, name, department, position, salary)
VALUES (11, '장미희', '기획', '사원', 3400000);
```

문제 4

'개발' 부서 직원의 이름과 급여를 조회하세요.

정답:

```sql
SELECT name, salary
FROM employees
WHERE department = '개발';
```

문제 5

'팀장' 직책을 가진 직원의 이름과 부서를 조회하세요.

정답:

```sql
SELECT name, department
FROM employees
WHERE position = '팀장';
```

※ 이후 문제 6 ~ 7 번은 안전모드를 해제하고 실행한 후 다시 안전모드를 활성하세요.

문제 6

'영업' 부서의 모든 직원 급여를 300,000원씩 인상하세요.

정답:

```sql
-- 안전 모드 해제
SET SQL_SAFE_UPDATES = 0;

-- 쿼리 실행
UPDATE employees
SET salary = salary + 300000
WHERE department = '영업';
```

문제 **7**

급여가 3,000,000원 이하인 직원을 삭제하세요.

정답:

```
-- 쿼리 실행
DELETE FROM employees
WHERE salary <= 3000000;

-- 안전 모드 다시 활성화
SET SQL_SAFE_UPDATES = 1;
```

Day 4

3장 데이터 필터링하기

오늘은 필요한 데이터만 골라내는 데이터 필터링에 대해 알아봅니다.

🧐 공부할 내용(89-108p)

- 데이터 필터링이란
- 데이터 필터링 실습: 대학 DB

1. WHERE 절과 주요 연산자

데이터 필터링(data filtering)이란 원하는 데이터만 걸러내는 작업으로 주로 WHERE 절을 통해 수행합니다. WHERE 절에 조건을 작성해 데이터 필터링할 때는 비교 연산자, 논리 연산자, 산술 연산자를 사용할 수 있습니다.

- 비교 연산자: 두 값을 비교하는 연산자로 =(같다), !=(같지 않다), >(크다), >=(크거나 같다), <(작다), <=(작거나 같다)가 있습니다.

- 논리 연산자: 두 조건을 조합해 새로운 조건을 만드는 연산자로 AND, OR, NOT이 있습니다. AND는 두 조건의 교집합을, OR는 합집합을, NOT은 여집합을 찾아줍니다.

- 산술 연산자: 사칙 연산을 위한 연산자로 +(더하기), -(빼기), *(곱하기), /(나누기), %(나머지)가 있습니다.

연산자 우선순위란 여러 연산자가 한 쿼리문에 있을 때 어떤 연산자를 먼저 수행할지 우선순위를 정한 것입니다. 가장 높은 우선순위부터 나열하면 다음과 같습니다.

- ()(괄호) → 논리 연산자 NOT → 산술 연산자(*, /, %) → 산술 연산자(+, -) → 비교 연산자(=, !=, >, >=, <, <=) → 논리 연산자 AND → 논리 연산자 OR

2. 연산자 확인 문제

다음 문제를 풀며 연산자를 제대로 이해했는지 확인해 봅시다.

문제 1

다음 쿼리의 결과로 조회되는 데이터는 무엇입니까까?

```
SELECT name, price
FROM products
WHERE price > 100000 AND price <= 500000;
```

① 가격이 100,000원 초과 500,000원 이하인 제품

② 가격이 100,000원 이상 500,000원 미만인 제품

③ 가격이 100,000원 이하 500,000원 초과인 제품

④ 가격이 100,000원 미만 500,000원 이하인 제품

정답: ①

문제 2

다음 쿼리의 결과로 조회되는 데이터는 무엇입니까?

```
SELECT id, name
FROM products
WHERE stock != 0;
```

① 재고(stock)가 0인 제품만 조회된다.

② 재고(stock)가 0이 아닌 제품만 조회된다.

③ 모든 제품이 조회된다.

④ 재고(stock)가 NULL인 제품만 조회된다.

정답: ②

문제 3

다음 쿼리를 실행했을 때 조회되는 데이터는 무엇입니까?

```
SELECT name, price, stock
FROM products
WHERE price < 20000 OR price > 50000 AND stock <= 20;
```

① 가격이 50,000원을 초과하고 재고가 20개 이하인 제품

② 가격이 20,000원을 초과하거나, 재고가 20개 이하인 제품

③ 가격이 50,000원을 초과하고 재고가 20개 이하이거나, 가격이 20,000원 미만인 제품

④ 가격이 50,000원을 초과하고 재고가 20개 초과인 제품

정답: ③(AND가 OR보다 우선순위가 높으므로 (price > 50000 AND stock <= 20) 조건이 먼저 평가된 후, 결과를 OR price < 20000 조건과 결합합니다.)

문제 4

다음 쿼리의 결과를 올바르게 해석한 것은 무엇입니까?

```
SELECT name, category
FROM products
WHERE (category = '전자기기' OR category = '가구') AND price >= 50000;
```

① 카테고리가 '전자기기' 또는 '가구'인 모든 제품

② 카테고리가 '전자기기' 또는 '가구'이고, 가격이 50,000원 이상인 제품

③ 가격이 50,000원 이상인 제품 중 카테고리가 '전자기기' 또는 '가구'가 아닌 제품

④ 카테고리가 '전자기기' 또는 '가구'이거나, 가격이 50,000원 이상인 제품

정답: ②(OR이 괄호로 묶여 있으므로 (category = '전자기기' OR category = '가구')가 먼저 평가되고, 그 결과에서 AND price >= 50000 조건이 적용됩니다.)

문제 **5**

다음 쿼리에서 WHERE 조건을 만족하는 데이터는 무엇입니까?

```
SELECT name, price, stock
FROM products
WHERE (price * 0.9 > 30000 OR stock <= 20) AND NOT (price < 10000);
```

① 할인 후 가격(price * 0.9)이 30,000원 초과하거나 재고가 20개 이하이면서, 가격이 10,000원 미만인 제품

② 할인 후 가격(price * 0.9)이 30,000원 초과하거나 재고가 20개 이하이면서, 가격이 10,000원 이상인 제품

③ 할인 후 가격(price * 0.9)이 30,000원 이하이고, 재고가 20개 초과이며, 가격이 10,000원 미만인 제품

④ 할인 후 가격(price * 0.9)이 30,000원 이하이고, 재고가 20개 초과하며, 가격이 10,000원 이상인 제품

정답: ②(price * 0.9 > 30000과 stock <= 20가 OR로 먼저 계산되고, NOT 조건이 price < 10000을 부정하여 price >= 10000로 평가됩니다.)

Day 5

3장 데이터 필터링하기

오늘은 데이터 필터링 문제를 풉니다.

🧐 공부할 내용(109p)

- 셀프체크

셀프체크의 정답을 모두 맞쳤나요? 그렇다면 추가 연습 문제를 풀며 실력을 향상해 봅시다.

1. 데이터 필터링 연습 문제

다음 products 테이블 생성과 데이터 삽입 쿼리를 보고 문제에 답하세요.

```
-- products 테이블 생성
CREATE TABLE products (
  id INTEGER,             -- ID
  name VARCHAR(50),       -- 제품명
  category VARCHAR(30),   -- 카테고리
  price INTEGER,          -- 가격
  stock INTEGER,          -- 재고
  PRIMARY KEY (id)
);
-- products 데이터 삽입
INSERT INTO products (id, name, category, price, stock)
VALUES
(1, '노트북', '전자기기', 1200000, 10),
(2, '스마트폰', '전자기기', 800000, 15),
(3, '청소기', '생활용품', 150000, 8),
(4, '텀블러', '생활용품', 12000, 50),
(5, '초코바', '식품', 1500, 100),
(6, '커피', '식품', 4500, 200),
(7, '에어컨', '전자기기', 1200000, 5),
(8, '책상', '가구', 50000, 20),
```

```
(9, '의자', '가구', 40000, 25),
(10, '모니터', '전자기기', 300000, 12);
```

문제 1

가격이 300,000원 이상인 제품명과 가격을 조회하세요.

정답:

```
SELECT name, price
FROM products
WHERE price >= 300000;
```

문제 2

카테고리가 '전자기기'이고 재고가 10개 이상인 제품명과 재고를 조회하세요.

정답:

```
SELECT name, stock
FROM products
WHERE category = '전자기기' AND stock >= 10;
```

문제 3

가격에 10% 세금을 적용한 최종 가격을 계산해 제품명과 함께 조회하세요.

정답:

```
SELECT name, price * 1.1 AS final_price
FROM products;
```

문제 4

카테고리가 '전자기기'가 아닌 제품을 찾아 제품명과 카테고리를 조회하세요.

정답:

```
SELECT name, category
FROM products
WHERE NOT category = '전자기기';
```

문제 5

재고가 10개 이하인 제품 중 가격을 20% 할인해 제품명과 가격을 조회하세요.

정답:

```
SELECT name, price * 0.8 AS discounted_price
FROM products
WHERE stock <= 10;
```

문제 6

카테고리가 '생활용품'이고 가격이 100,000원 이상이거나, 재고가 50개 이상인 제품의 제품명, 카테고리, 재고를 조회하세요.

정답:

```
SELECT name, category, stock
FROM products
WHERE (category = '생활용품' AND price >= 100000) OR stock >= 50;
```

문제 7

카테고리가 '전자기기'인 제품 중 재고가 10개 이하 남은 제품을 30% 할인된 가격으로 판매하려고 합니다. 해당 제품의 제품명, 재고, 할인된 가격을 조회하세요.

정답:

```
SELECT name, stock, price * 0.7 AS final_price
FROM products
WHERE category = '전자기기' AND stock <= 10;
```

문제 8

각 제품의 재고를 모두 소진했을 때 매출을 구해 제품명과 총판매액을 출력하세요.

정답:

```
SELECT name, price * stock AS total_sales_amount
FROM products
```

Day 6

4장 데이터 집계하기

오늘은 집계 함수를 이용해 데이터를 집계하는 방법을 알아봅니다.

🧐 공부할 내용(111~128p)

- 집계 함수란
- 집계 함수 실습: 은행 DB

1. 집계 함수란

집계 함수는 특정 칼럼 값을 입력받아 통계적 계산을 해 주는 함수로 최댓값, 최솟값, 합계, 평균 등을 구할 때 사용합니다. 책에서는 최댓값을 반환하는 MAX(), 최솟값을 반환하는 MIN(), 튜플의 개수를 반환하는 COUNT(), 합계를 반환하는 SUM(), 평균을 반환하는 AVG()에 대해 알아봤습니다.

집계 함수의 사용 형식은 다음과 같습니다.

```
SELECT 집계함수명(칼럼명)
FROM 테이블명;
```

2. 집계 함수 확인 문제

다음 문제를 풀며 집계 함수를 제대로 이해했는지 확인해 봅시다.

문제 **1**

다음 쿼리의 결과로 올바르게 반환되는 값은 무엇입니까?

```sql
SELECT MAX(price) AS highest_price
FROM products
WHERE category = '전자기기';
```

① 모든 제품의 가장 높은 가격

② 전자기기 카테고리에서 가장 높은 가격

③ 전자기기 카테고리에서 가장 낮은 가격

④ 모든 제품의 가장 낮은 가격

정답: ②

문제 **2**

COUNT() 함수에 대한 설명 중 옳지 <u>않은</u> 것은?

① COUNT(*)는 NULL 값을 포함한 전체 튜플의 수를 센다.

② COUNT(category)는 NULL 값을 포함한 카테고리의 수를 센다.

③ COUNT(DISTINCT category)는 중복을 제외한 카테고리의 수를 센다.

④ 모든 카테고리 값이 다르다면, COUNT(DISTINCT category)로 조회하던 COUNT(category) 조회하던 결과는 같다(NULL 값은 없다고 가정한다).

정답: ②

문제 3

다음 쿼리의 결과를 올바르게 해석한 것은 무엇입니까?

```
SELECT SUM(stock) AS total_stock
FROM products
WHERE NOT (price > 10000);
```

① 가격이 10,000원을 초과하는 제품의 재고

② 가격이 10,000원을 초과하는 제품의 재고 총합

③ 가격이 10,000원 이하인 제품의 재고

④ 가격이 10,000원 이하인 제품의 재고 총합

정답: ④

문제 4

다음 쿼리의 실행 결과는 무엇입니까?

```
SELECT AVG(price) AS average_price
FROM products
WHERE category IN ('전자기기', '생활용품');
```

① 전자기기와 생활용품 카테고리의 평균 가격

② 모든 제품의 평균 가격

③ 전자기기 또는 생활용품 카테고리의 최대 가격

④ 전자기기 또는 생활용품 카테고리의 가격 합계

정답: ①

문제 **5**

다음 쿼리의 결과는 무엇을 나타냅니까?

```sql
SELECT MIN(stock) AS minimum_stock
FROM products
WHERE price >= 20000 AND price <= 100000;
```

① 모든 제품의 최소 재고

② 가격이 20,000원 이상 100,000원 이하인 제품의 최소 재고

③ 가격이 20,000원 이상인 제품의 최소 재고

④ 가격이 100,000원 이하인 제품의 최소 재고

정답: ②

문제 **6**

다음 쿼리의 결과로 반환되는 데이터는 무엇입니까?

```sql
SELECT COUNT(*) AS total_products
FROM products
WHERE rating IN (4.0, 4.5);
```

① 모든 제품의 개수

② 평점(rating)이 4.0 또는 4.5인 제품의 개수

③ 평점(rating)이 4.0부터 4.5인 제품의 개수

④ 평점(rating)이 4.0과 4.5가 아닌 제품의 개수

정답: ②

문제 7

다음 쿼리의 조건을 올바르게 해석하세요.

```sql
SELECT MAX(price) AS highest_price
FROM products
WHERE stock >= 10 AND category IN ('식품', '전자기기');
```

① 모든 제품의 최대 가격

② 식품 또는 전자기기의 최대 가격

③ 재고가 10 이상인 식품과 전자기기의 최대 가격

④ 재고가 10 미만인 식품과 전자기기의 최대 가격

정답: ③

Day 7

4장 데이터 집계하기

오늘은 집계 함수에 관한 문제를 풉니다.

🧐 공부할 내용(129p)

- 셀프체크

셀프체크의 정답을 모두 맞쳤나요? 그렇다면 추가 연습 문제를 풀며 실력을 향상해 봅시다.

1. 집계 함수 연습 문제

다음 orders(주문) 테이블을 보고 문제에 답하세요.

```
-- orders 테이블 생성
CREATE TABLE orders (
  id INTEGER,                    -- ID
  customer_name VARCHAR(50),     -- 고객명
  product VARCHAR(50),           -- 제품명
  quantity INTEGER,              -- 주문 수량
  price DECIMAL(10, 2),          -- 단가
  order_date DATE,               -- 주문 날짜
  region VARCHAR(20),            -- 고객이 사는 지역
  PRIMARY KEY (id)
);
-- orders 데이터 삽입
INSERT INTO orders (id, customer_name, product, quantity, price,
order_date, region)
VALUES
(1, '김철수', '노트북', 2, 1500000, '2023-11-01', '서울'),
(2, '박영희', '스마트폰', 1, 900000, '2023-11-02', '부산'),
(3, '이민호', '청소기', 1, 250000, '2023-11-03', '서울'),
(4, '최수진', '냉장고', 1, 1200000, '2023-11-04', '대구'),
(5, '정하늘', '노트북', 1, 1500000, '2023-11-05', '부산'),
```

```
(6, '홍길동', '스마트폰', 3, 900000, '2023-11-06', '서울'),
(7, '오준수', '에어컨', 2, 800000, '2023-11-07', '대구'),
(8, '서지우', '청소기', 1, 250000, '2023-11-08', '서울'),
(9, '이은지', '냉장고', 2, 1200000, '2023-11-09', '부산'),
(10, '안현준', '스마트폰', 1, 900000, '2023-11-10', '대구');
```

문제 1

모든 주문의 총 매출액을 계산하세요.

정답:

```
SELECT SUM(quantity * price) AS total_sales
FROM orders;
```

문제 2

단가가 1,000,000원 이상인 제품의 주문 건수를 계산하세요.

정답:

```
SELECT COUNT(*) AS expensive_orders
FROM orders
WHERE price >= 1000000;
```

문제 3

주문 수량이 2개 이상이면서 단가가 1,000,000원 이하인 제품의 총 매출액을 계산하세요.

정답:

```
SELECT SUM(quantity * price) AS total_sales
FROM orders
WHERE quantity >= 2 AND price <= 1000000;
```

문제 4

고객이 사는 지역(region)의 개수를 출력하세요.

```
SELECT COUNT(DISTINCT region) AS unique_regions
FROM orders;
```

문제 5

주문 날짜가 2023-11-01과 2023-11-05 사이에 해당하는 주문의 총 수량을 계산하세요.

정답:

```sql
SELECT SUM(quantity) AS total_quantity
FROM orders
WHERE order_date >= '2023-11-01' AND order_date <= '2023-11-05';
```

문제 6

고객 이름이 '김철수'이거나 '대구'에 사는 고객이 주문한 제품명을 조회하세요.

정답:

```sql
SELECT DISTINCT product AS unique_products
FROM orders
WHERE customer_name = '김철수' OR region = '대구';
```

문제 7

'스마트폰'이 총 몇 대 팔렸는지 조회하세요.

정답:

```sql
SELECT SUM(quantity)
FROM orders
WHERE product = '스마트폰';
```

Day 8

5장 다양한 자료형 활용하기

오늘은 칼럼의 성격에 맞게 자료형을 지정하는 방법을 알아봅니다.

🧐 공부할 내용**(131~148p)**

- 자료형이란

1. 자료형이란

자료형이란 데이터의 형태를 말하는 것으로 숫자형(정수형, 실수형), 문자형, 날짜 및 시간형 등이 있습니다.

- 정수형: 2, -1, 0, 1, 2, ...와 같이 소수점이 없는 숫자를 저장합니다. 차지하는 메모리 크기에 따라 TINYINT(매우 작은 정수 저장), SMALLINT(작은 정수 저장), MEDIUMINT(중간 정수 저장), INTEGER 또는 INT(표준 정수 저장), BIGINT(매우 큰 정수 저장)의 세부 타입을 선택해 사용할 수 있습니다.

- 실수형: 소수점이 있는 실수를 저장합니다. 소수점의 위치가 변하는 실수를 저장하는 부동 소수점 방식의 FLOAT와 DOUBLE 타입과, 소수점 이하 자릿수가 고정된 실수를 저장하는 고정 소수점 방식의 DECIMAL(P, S) 타입이 있습니다.

- 문자형: 한글, 영어, 기호, 문자화한 숫자 등을 저장하기 위한 자료형입니다. 저장하는 문자의 길이에 따라 CHAR, VARCHAR, TEXT, BLOB, ENUM 등의 세부 타입이 있습니다.

- 날짜 및 시간형: 날짜 및 시간형은 말 그대로 날짜와 시간 값을 저장하기 위한 자료형으로 DATE, TIME, DATETIME, YEAR의 세부 타입이 있습니다.

2. 자료형 확인 문제

자료형에 대한 문제를 읽고 답하세요(주관식).

문제 1

다음 쿼리에서 id 칼럼의 자료형으로 SMALLINT가 적합한 이유를 설명하세요.

```
CREATE TABLE employees (
  id SMALLINT UNSIGNED,
  age TINYINT,
  salary INTEGER,
  PRIMARY KEY (id)
);
```

정답:

SMALLINT는 2바이트로 메모리를 절약하면서 65,535까지의 정수를 저장할 수 있기 때문이다.

문제 2

다음 쿼리에서 DECIMAL(5, 2) 자료형의 의미는 무엇입니까?

```
CREATE TABLE products (
  id INTEGER,
  price DECIMAL(5, 2),
  ...
);
```

정답:

최대 5자리 실수를 저장하며, 소수점 이하 2자리를 포함한다.

문제 3

다음 쿼리에서 VARCHAR(50)과 CHAR(50)의 주요 차이점은 무엇입니까?

```
CREATE TABLE users (
  username VARCHAR(50),
  password CHAR(50),
  ...
);
```

정답:

VARCHAR(50)과 CHAR(50)은 문자 50자를 저장할 수 있는 공간을 제공한다. 그러나 작동 방식이 다르다.

- VARCHAR(50): 최대 50자의 가변 길이 문자열을 저장할 수 있으며, 실제 저장된 문자열의 길이에 따라 메모리를 사용해 공간 낭비를 줄인다.
- CHAR(50): 최대 50자의 고정 길이 문자열을 저장하며, 데이터 길이와 상관없이 항상 50자 공간을 사용한다.

문제 4

다음 쿼리에서 DATETIME과 DATE의 주요 차이점은 무엇입니까?

```
CREATE TABLE events (
  event_id INTEGER,
  start_time DATETIME,
  event_date DATE,
  ...
);
```

정답:

DATETIME은 날짜와 시간을 모두 저장하고, DATE는 날짜만 저장한다.

문제 5

다음 쿼리에서 ENUM 자료형이 사용된 이유를 설명하세요.

```
CREATE TABLE employees (
  id INTEGER,
  gender ENUM('M', 'F'),
  ...
);
```

정답:

ENUM은 제한된 값 목록만 저장하도록 강제할 수 있기 때문이다.

문제 6

다음 쿼리에서 BLOB 자료형은 어떤 데이터를 저장하는 데 적합합니까?

```
CREATE TABLE files (
  file_id INTEGER,
  file_data BLOB NOT NULL,
  ...
);
```

정답:

이미지, 오디오, 동영상 같은 크기가 큰 파일을 저장하는 데 적합하다.

문제 7

다음 쿼리에서 FLOAT와 DOUBLE의 차이점을 설명하세요.

```
CREATE TABLE measurements (
  id INTEGER,
  value FLOAT,
  precise_value DOUBLE,
  ...
);
```

정답:

FLOAT와 DOUBLE은 둘 다 부동 소수점 방식의 실수를 저장한다. 다만 차지하는 메모리 크기가 달라 FLOAT는 4바이트, DOUBLE은 8바이트를 차지한다. 따라서 DOUBLE은 FLOAT보다 넓은 범위의 실수를 저장해 실수를 좀 더 정밀하게 표현할 수 있다.

문제 8

다음 쿼리에서 DATE 자료형이 저장할 수 있는 값의 범위는 무엇입니까?

```
CREATE TABLE schedules (
  id INTEGER,
  meeting_date DATE,
  ...
);
```

정답:

1000-01-01부터 9999-12-31이다.

Day 9

5장 다양한 자료형 활용하기

오늘은 자료형에 따른 필터링 실습을 합니다.

🧐 공부할 내용(149~162p)

- 자료형에 따른 필터링 실습: 상점 DB

1. LIKE 연산자, 날짜/시간 함수, BETWEEN 연산자

- **LIKE**: 칼럼 값이 특정 패턴과 완전히 일치하거나 특정 패턴을 포함하는지 확인할 때 사용하는 연산자입니다. 와일드 카드(%, _)와 함께 사용하면 특정 패턴이 칼럼 값으로 포함됐는지 확인할 수 있습니다.

- 날짜 함수: 입력받은 날짜 데이터에서 연도, 월, 일을 추출할 때 사용하는 함수로, YEAR(날짜), MONTH(날짜), DAY(날짜), EXTRACT(필드 FROM 날짜) 등이 있습니다.

- 시간 함수: 입력받은 시간 데이터에서 시, 분, 초 등을 추출할 때 사용하는 함수로, HOUR(시간), MINUTE(시간), SECOND(시간), TIME_TO_SEC(시간) 등이 있습니다.

- **BETWEEN**: 두 값 사이에 속하는지 확인할 때 사용하는 연산자입니다. BETWEEN 연산의 결과로 시작 값과 마지막 값을 포함해 두 값 사이에 속하는 튜플을 필터링합니다.

2. LIKE 연산자, 날짜/시간 함수, BETWEEN 연산자 확인 문제

다음 문제를 읽고 답하세요(주관식 문제).

문제 1

LIKE 연산자를 사용할 때 _(언더스코어)의 역할은 무엇입니까?

정답:

LIKE 연산자에서 _는 정확히 한 글자를 대체한다.

문제 2

다음 WHERE 절이 의미하는바를 설명하세요.

```
WHERE customer_name LIKE '박__'
```

정답:

고객 이름이 '박'으로 시작하며 3글자를 찾는다.

문제 3

날짜에서 월을 추출하는 함수는 무엇입니까?

정답:

MONTH()

문제 4

다음 쿼리의 결과는 무엇입니까?

```
SELECT EXTRACT(DAY FROM '2024-01-15');
```

정답:

15 (EXTRACT(DAY FROM date) 함수는 날짜에서 일(Day)을 추출합니다.)

문제 5

다음 쿼리의 결과는 무엇입니까?

```
SELECT HOUR('15:45:20');
```

정답:

15 (HOUR() 함수는 시간에서 시간(Hour)을 추출합니다.)

문제 6

다음 쿼리의 조건에 해당하는 고객 이름을 [보기]에서 모두 고르세요.

```
WHERE customer_name LIKE '%현%';
```

```
[보기] 안현준, 현철수, 박영희, 정수현, 김길벗
```

정답:

안현준, 현철수, 정수현

문제 7

다음 WHERE 절의 조건이 의미하는바를 설명하세요.

```
WHERE HOUR(order_time) BETWEEN 9 AND 17
```

정답:

주문 시간이 9시부터 17시까지인 주문을 조회한다.

Day 10

5장 다양한 자료형 활용하기

오늘은 자료형에 따른 필터링 문제를 풉니다.

🧐 공부할 내용(163p)

- 셀프체크

셀프체크의 정답을 모두 맞쳤나요? 그렇다면 추가 연습 문제를 풀며 실력을 향상해 봅시다.

1. 자료형에 따른 필터링 연습 문제

다음 이벤트(events) 테이블을 보고 문제에 답하세요.

```
-- events 테이블 생성
CREATE TABLE events (
  id INTEGER,            -- ID
  title VARCHAR(100),    -- 이벤트 제목
  event_date DATE,       -- 이벤트 날짜
  start_time TIME,       -- 시작 시간
  location VARCHAR(50), -- 장소
  attendees INT,         -- 참석자 수
  PRIMARY KEY (id)
);
-- events 데이터 삽입
INSERT INTO events (id, title, event_date, start_time, location,
attendees)
VALUES
(1, '코딩 부트캠프', '2023-10-01', '09:00:00', '서울', 50),
(2, 'AI 세미나', '2023-10-15', '14:00:00', '부산', 100),
(3, '데이터 분석 워크숍', '2023-11-05', '10:30:00', '서울', 30),
(4, '스타트업 데모데이', '2023-12-10', '13:00:00', '대전', 200),
(5, '클라우드 컨퍼런스', '2024-01-20', '11:15:00', '인천', 150),
(6, '해커톤', '2024-02-05', '08:00:00', '서울', 300),
```

```
(7, 'UX/UI 디자인 워크숍', '2023-09-25', '09:30:00', '광주', 25),
(8, '기술 트렌드 토크', '2023-11-20', '15:00:00', '서울', 80),
(9, '프로그래밍 대회', '2023-12-01', '10:00:00', '부산', 120),
(10, '오픈소스 컨트리뷰션 데이', '2023-10-25', '16:00:00', '서울', 60);
```

문제 1

이벤트 제목에 '워크숍'이 포함된 이벤트의 제목과 장소를 조회하세요.

정답:

```
SELECT title, location
FROM events
WHERE title LIKE '%워크숍%';
```

문제 2

이벤트 제목이 '데이터'로 시작하는 이벤트의 제목과 참석자 수를 조회하세요.

정답:

```
SELECT title, attendees
FROM events
WHERE title LIKE '데이터%';
```

문제 3

이벤트 날짜가 2023년인 이벤트의 제목과 날짜를 조회하세요.

정답:

```
SELECT title, event_date
FROM events
WHERE YEAR(event_date) = 2023;
```

문제 4

이벤트 시작 시간이 오전 9시 이전인 이벤트의 제목과 시작 시간을 조회하세요.

정답:

```
SELECT title, start_time
FROM events
WHERE HOUR(start_time) < 9;
```

문제 5

참석자가 50명 이상 150명 이하인 이벤트의 제목과 참석자 수를 조회하세요.

정답:

```
SELECT title, attendees
FROM events
WHERE attendees BETWEEN 50 AND 150;
```

문제 6

이벤트 날짜가 2023-10-01부터 2023-12-31 사이인 이벤트의 제목과 날짜를 조회하세요.

정답:

```
SELECT title, event_date
FROM events
WHERE event_date BETWEEN '2023-10-01' AND '2023-12-31';
```

문제 7

이벤트 제목에 '컨퍼런스' 또는 '컨트리뷰션'이 포함되고, 시작 시간이 오전 11시 이후인 이벤트의 제목과 시작 시간을 조회하세요.

정답:

```
SELECT title, start_time
FROM events
WHERE (title LIKE '%컨퍼런스%' OR title LIKE '%컨트리뷰션%')
      AND HOUR(start_time) >= 11;
```

문제 **8**

이벤트 날짜가 2023-11-1부터 2023-12-31 사이고, 시작 시간이 오후 2시 이후인 이벤트의 제목과 날짜, 시작 시간을 조회하세요.

정답:

```
SELECT title, event_date, start_time
FROM events
WHERE event_date BETWEEN '2023-11-01' AND '2023-12-31'
      AND HOUR(start_time) >= 14;
```

Day 11

6장 관계 만들기

오늘은 테이블 간에 관계를 만드는 방법을 알아봅니다.

🧐 공부할 내용**(165-183p)**

- 관계란
- 다양한 관계 만들기

1. 관계의 의미

관계란 여러 테이블에 분산 저장된 데이터가 서로 어떻게 연관돼 있는지를 정의하는 개념입니다. 관계의 종류로는 일대일 관계, 일대다 관계, 다대다 관계가 있습니다.

- 일대일 관계: 하나의 데이터가 하나의 데이터와만 연결된 관계입니다. 한 사람은 한 주민등록증을 가지고, 한 주민등록증은 한 사람에게만 발급되므로 '사람'과 '주민등록증'은 일대일 관계입니다.

- 일대다 관계: 하나의 데이터가 여러 데이터와 연결된(포함 또는 소유) 관계입니다. 한 회사는 여러 직원을 고용할 수 있지만, 한 직원은 한 회사에만 소속되므로 '회사'와 '직원'은 일대다 관계입니다.

- 다대다 관계: 여러 데이터가 여러 데이터와 연결된 관계입니다. 한 의사는 여러 환자를 진료할 수 있고, 한 환자는 여러 의사에게 진료를 받을 수 있으므로 '의사'와 '환자'는 다대다 관계입니다.

2. 관계를 만드는 방법

기본키와 외래키

기본키(Primary Key)는 테이블에서 각 행을 고유하게 식별할 수 있는 칼럼 또는 칼럼의 집합을 말합니다. 기본키는 값이 중복되면 안 되고, NULL 값이 와도 안 됩니다.

외래키(Foreign Key)는 다른 테이블의 기본키를 참조해 두 테이블 간의 관계를 설정하는 칼럼입니다. 외래키는 반드시 다른 테이블의 기본키 값만 가질 수 있으며, 이를 통해 두 테이블 간의 데이터 일관성을 보장합니다.

```
CREATE TABLE 테이블명 (
  칼럼명 자료형,
  ...
  PRIMARY KEY (칼럼명),
  FOREIGN KEY (칼럼명) REFERENCES 연결_테이블(연결_테이블의_기본키)
);
```

제약조건

칼럼에 저장할 수 있는 값에 대해 어떤 제한이나 규칙을 거는 것을 말합니다.

- **AUTO_INCREMENT**: 칼럼의 값을 자동으로 1씩 증가시킵니다.
- **UNIQUE**: 칼럼의 값으로 고유한 값만 허용합니다(중복되면 안 됨).
- **NOT NULL**: 칼럼에 NULL을 허용하지 않습니다.
- **DEFAULT**: 칼럼에 기본값을 지정합니다.
- **CHECK**: 칼럼의 값이 특정 조건을 만족하도록 조건을 지정합니다.
- **UNSIGNED**: 음수를 허용하지 않고 양수 값만 저장하도록 제한합니다.

```
CREATE TABLE products (
  id INTEGER AUTO_INCREMENT,          -- 상품 ID(자동 증가)
  name VARCHAR(100) UNIQUE,           -- 상품명(고유한 값만 허용)
  category VARCHAR(50) NOT NULL,      -- 상품 카테고리(NULL 불가)
  status VARCHAR(20) DEFAULT 'available',-- 상품 상태(기본값: available)
  dc_rate INTEGER CHECK (dc_rate BETWEEN 0 AND 50), -- 할인율(0~50% 제한)
  stock INTEGER UNSIGNED,             -- 재고 수량(음수 불가)
  PRIMARY KEY (id)                    -- 기본키 설정
);
```

일대일 관계 맺는 법

한 테이블의 기본키를 다른 테이블의 외래키로 설정하고, 외래키에 UNIQUE 제약조건을 추가합니다.

예: 사람과 여권(한 사람은 한 개의 여권을 가지고, 한 여권은 한 사람에게만 발급)

```
-- 사람 테이블
CREATE TABLE persons (
  id INTEGER,         -- ID
  name VARCHAR(50),   -- 이름
  PRIMARY KEY (id)    -- 기본키 지정: id
);
-- 여권 테이블
CREATE TABLE passports (
  id INTEGER,                   -- ID
  passport_number VARCHAR(20), -- 여권 번호
  person_id INTEGER UNIQUE,     -- 사람_ID
  PRIMARY KEY (id),             -- 기본키 지정: id
  FOREIGN KEY (person_id) REFERENCES persons(id) -- 외래키 지정: person_id
);
```

일대다 관계 맺는 법

'일' 쪽의 테이블의 기본키를 '다' 쪽 테이블의 외래키로 설정합니다.

예: 회사와 직원(한 회사는 여러 직원을 고용할 수 있지만, 한 직원은 하나의 회사에만 소속)

```
-- 회사 테이블
CREATE TABLE companies (
  id INTEGER,        -- ID
  name VARCHAR(50), -- 회사명
  PRIMARY KEY (id)  -- 기본키 지정: id
);
-- 직원 테이블
CREATE TABLE employees (
  id INTEGER,              -- ID
  name VARCHAR(50),        -- 직원명
  company_id INTEGER,      -- 회사_ID
  PRIMARY KEY (id),        -- 기본키 지정: id
  FOREIGN KEY (company_id) REFERENCES companies(id) -- 외래키 지정: company_id
);
```

다대다 관계 맺는 법

중간 테이블을 만들어 두 테이블의 기본키를 외래키로 설정합니다.

예: 학생과 과목(한 학생은 여러 과목을 수강하고, 한 과목은 여러 학생이 수강할 수 있음)

```
-- 학생 테이블
CREATE TABLE students (
  id INTEGER,           -- ID
  name VARCHAR(50),     -- 학생명
  PRIMARY KEY (id)      -- 기본키 지정: id
);
-- 과목 테이블
CREATE TABLE subjects (
  id INTEGER,           -- ID
  title VARCHAR(50),    -- 과목명
  PRIMARY KEY (id)      -- 기본키 지정: id
);
-- 수강 테이블(중간 테이블)
CREATE TABLE enrollments (
  id INTEGER,           --  ID
  student_id INTEGER,   -- 학생_ID
  subject_id INTEGER,   -- 과목_ID
  PRIMARY KEY (id),     -- 기본키 지정: id
  FOREIGN KEY (student_id) REFERENCES students(id), -- 외래키 지정: student_id
  FOREIGN KEY (subject_id) REFERENCES subjects(id)  -- 외래키 지정: subject_id
);
```

3. 관계 확인 문제

'관계의 개념'과 '관계 맺는 법'을 잘 이해했는지 문제를 풀며 확인해 봅시다.

문제 1

다음 중 일대다 관계에 해당하는 예는 무엇입니까?

① 국가와 수도　　② 부모와 자식

③ 학생과 수강 과목　　④ 책과 ISBN 번호

정답: ②

문제 **2**

AUTO_INCREMENT 제약 조건에 대한 설명으로 옳은 무엇입니까?

① 특정 칼럼의 값이 항상 고유해야 함을 보장한다.

② 특정 컬럼의 값이 NULL을 허용하지 않도록 제한한다.

③ 새 행이 삽입될 때마다 값을 1씩 자동으로 증가한다.

④ 칼럼 값이 음수를 허용하지 않도록 제한한다.

정답: ③

문제 **3**

다음 중 CHECK 제약 조건이 설정된 칼럼에 대해 위배되는 입력값을 고르세요.

```
discount_rate INTEGER CHECK (discount_rate >= 0 AND discount_rate <= 50)
```

① -5 ② 0 ③ 25 ④ 50

정답: ①

문제 **4**

다음 중 UNIQUE 제약 조건을 사용하는 이유로 가장 적합한 것은 무엇입니까?

① 특정 칼럼의 기본값을 설정하기 위해

② 특정 칼럼에 NULL 값을 허용하지 않기 위해

③ 특정 칼럼 값이 중복되지 않도록 보장하기 위해

④ 칼럼 값이 자동으로 증가하도록 설정하기 위해

정답: ③

Day 12

6장 관계 만들기

오늘은 관계 만들기 실습과 셀프체크를 진행합니다.

🧐 공부할 내용(184-200p)

- 관계 만들기 실습: 별그램 DB
- 셀프체크

셀프체크의 정답을 모두 맞쳤나요? 그렇다면 추가 연습 문제를 풀며 실력을 향상해 봅시다.

1. 관계 만들기 연습 문제

도서관 데이터베이스는 members(회원), member_profiles(회원 프로필), books(도서), borrow_records(대출 기록), library_staff(도서관 직원)의 5개 테이블로 구성됩니다. 코드를 보고 문제에 답하세요.

```
-- members 테이블
CREATE TABLE members (
  id INTEGER AUTO_INCREMENT,     -- ID(자동 증가)
  name VARCHAR(50) NOT NULL,     -- 회원명(NULL 불가)
  email VARCHAR(100) UNIQUE,     -- 이메일(고유 값)
  phone_number CHAR(15),         -- 전화번호
  membership_status VARCHAR(20) DEFAULT 'active', -- 회원 상태(기본값: active)
  PRIMARY KEY (id)               -- 기본키 지정: id
);
-- member_profiles 테이블
CREATE TABLE member_profiles (
  id INTEGER AUTO_INCREMENT,     -- ID(자동 증가)
  date_of_birth DATE,            -- 생년월일
  address TEXT,                  -- 주소
  member_id INTEGER UNIQUE,      -- 회원_ID(고유 값)
  PRIMARY KEY (id),              -- 기본키 지정: id
```

```
  FOREIGN KEY (member_id) REFERENCES members(id) -- 외래키 지정: member_id
);
-- books 테이블
CREATE TABLE books (
  id INTEGER AUTO_INCREMENT,          -- ID(자동 증가)
  title VARCHAR(100) NOT NULL,        -- 도서명(NULL 불가)
  author VARCHAR(100),                -- 저자
  category VARCHAR(50),               -- 카테고리
  stock INTEGER UNSIGNED DEFAULT 0,   -- 재고(음수 불가, 기본값: 0)
  PRIMARY KEY (id)                    -- 기본키 지정: id
);
-- borrow_records 테이블
CREATE TABLE borrow_records (
  id INTEGER AUTO_INCREMENT,          -- ID(자동 증가)
  borrow_date DATE NOT NULL,          -- 대출 날짜(NULL 불가)
  return_date DATE,                   -- 반납 날짜
  member_id INTEGER NOT NULL,         -- 회원_ID
  book_id INTEGER NOT NULL,           -- 도서_ID
  PRIMARY KEY (id),                   -- 기본키 지정: id
  FOREIGN KEY (member_id) REFERENCES members(id), -- 외래키 지정: member_id
  FOREIGN KEY (book_id) REFERENCES books(id)      -- 외래키 지정: book_id
);
-- library_staff 테이블
CREATE TABLE library_staff (
  id INTEGER AUTO_INCREMENT,          -- ID(자동 증가)
  name VARCHAR(50) NOT NULL,          -- 직원명(NULL 불가)
  role VARCHAR(50) DEFAULT 'staff',   -- 역할(기본값: staff)
  employment_date DATE NOT NULL,      -- 고용 날짜(NULL 불가)
  salary INTEGER UNSIGNED CHECK (salary >= 0), -- 급여(음수 불가)
  PRIMARY KEY (id)                    -- 기본키 지정: id
);
```

문제 1

members 테이블에서 email 칼럼에 적용된 제약 조건으로 올바른 것은 무엇입니까?

① AUTO_INCREMENT　　② UNIQUE

③ NOT NULL　　④ DEFAULT

정답: ②

문제 2

member_profiles 테이블에서 member_id 칼럼의 역할은 무엇입니까?

① 모든 회원 프로필을 구분하기 위한 기본키

② members 테이블의 기본키를 참조하는 외래키

③ 회원의 생년월일을 저장하는 칼럼

④ 회원의 이메일을 저장하는 칼럼

정답: ②

문제 3

books 테이블의 stock 칼럼에서 허용되지 않는 값은 무엇입니까?

① 100 ② 0 ③ -10 ④ NULL

정답: ③

문제 4

borrow_records 테이블에서 book_id와 연결된 테이블은 무엇입니까?

① books 테이블 ② members 테이블

③ member_profiles 테이블 ④ library_staff 테이블

정답: ①

문제 5

library_staff 테이블의 salary 칼럼에 대해 CHECK 제약 조건으로 허용되지 않는 값은 무엇입니까?

① 0 ② 100000

③ -5000 ④ 5000

정답: ③

문제 6

다음 중 AUTO_INCREMENT 제약 조건에 대한 설명으로 옳은 것은 무엇입니까?

① NULL 값을 허용하지 않음

② 행 삽입 시 값이 자동으로 1씩 증가

③ 다른 테이블의 값을 참조

④ 칼럼 값이 항상 고유하도록 설정

정답: ②

문제 7

borrow_records 테이블의 기본키는 무엇입니까?

① member_id

② book_id

③ id

④ borrow_date

정답: ③

문제 8

member_profiles 테이블의 member_id 칼럼에 UNIQUE 제약 조건을 적용한 이유는 무엇입니까?

① 회원이 여러 개의 프로필을 가질 수 있도록 허용하기 위해

② 회원 ID가 중복 입력되는 것을 허용하지 않기 위해

③ 회원 프로필과 도서 대출 기록을 연결하기 위해

④ 회원 테이블과의 관계를 유지하기 위해

정답: ④

문제 **9**

library_staff 테이블의 role 칼럼의 기본값은 무엇입니까?

① NULL ② 관리자

③ 직원 ④ staff

정답: ④

문제 **10**

borrow_records 테이블은 어떤 관계를 표현합니까?

① 회원과 도서 간의 일대일 관계

② 회원과 도서 간의 일대다 관계

③ 회원과 도서 간의 다대다 관계

④ 도서와 직원 간의 관계

정답: ③

Day 13

7장 테이블 조인하기

벌써 공부를 시작한 지 13일차네요. 오늘은 여러 테이블에 있는 데이터를 하나로 가져오는 방법인 조인에 대해 알아보겠습니다.

🧐 공부할 내용(201-218p)

- 조인이란
- 조인의 유형

1. 조인의 개념과 유형

조인(Join)은 2개 이상의 테이블을 연결하여 관련된 데이터를 하나의 결과 집합으로 가져오는 SQL 연산입니다. 조인은 테이블 간의 관계를 기반으로 데이터를 결합하여 의미 있는 정보를 제공합니다.

조인의 유형에는 INNER 조인, LEFT 조인, RIGHT 조인, FULL 조인이 있습니다.

INNER 조인

가장 기본이 되는 조인으로, 두 테이블에서 조인 조건을 만족하는 데이터를 찾아 연결합니다. INNER 키워드는 생략할 수 있습니다.

```
SELECT 칼럼명1, 칼럼명2, ...
FROM 테이블A
INNER JOIN 테이블B ON 테이블A.조인_칼럼 = 테이블B.조인_칼럼;
```

LEFT 조인

왼쪽 테이블(FROM 절의 테이블)의 모든 데이터에 대해 오른쪽 테이블(JOIN 절의 테이블)의 조인 조건을 만족하는 데이터를 찾아 조인하고, 조인이 불가능한 데이터는 NULL 값으로 채워 결과 테이블을 완성합니다.

```
SELECT 칼럼명1, 칼럼명2, ...
FROM 테이블A
LEFT JOIN 테이블B ON 테이블A.조인_칼럼 = 테이블B.조인_칼럼;
```

RIGHT 조인

오른쪽 테이블(JOIN 절의 테이블)의 모든 데이터에 대해 왼쪽 테이블(FROM 절의 테이블)의 조인 조건을 만족하는 데이터를 찾아 조인하고, 조인이 불가능한 데이터는 NULL 값으로 채워 결과 테이블을 완성합니다. LEFT 조인과 비교하면 기준 테이블만 바뀌었습니다.

```
SELECT 칼럼명1, 칼럼명2, ...
FROM 테이블A
RIGHT JOIN 테이블B ON 테이블A.조인_칼럼 = 테이블B.조인_칼럼;
```

FULL 조인

두 테이블 간 모든 튜플을 결합하는 조인입니다. 두 테이블에 조인 칼럼 값이 같은 데이터뿐만 아니라 한 테이블에만 존재하는 데이터도 모두 반환하고 빈 칼럼은 NULL 값으로 채웁니다.

```
SELECT 칼럼명1, 칼럼명2, ...
FROM 테이블A
FULL JOIN 테이블B ON 테이블A.조인_칼럼 = 테이블B.조인_칼럼;
```

MySQL에서는 FULL 조인을 지원하지 않습니다. 대신 UNION 연산자를 사용하면 FULL 조인한 결과와 같이 만들 수 있습니다.

UNION 연산자

두 쿼리의 결과 테이블을 하나로 합치는 집합 연산자로, 중복 튜플은 제거하고 합칩니다. 두 쿼리의 결과 테이블 내 칼럼 개수와 각 칼럼의 자료형이 정확히 일치해야 합니다.

```
(쿼리A)
UNION
(쿼리B);
```

2. 조인 확인 문제

문제 1

다음 중 조인의 정의로 가장 적절한 것은 무엇입니까?

① 2개 이상 테이블의 모든 데이터를 결합

② 2개 이상 테이블에서 관련된 데이터를 결합

③ 두 테이블 간 조건 없이 모든 행을 결합

④ 하나의 테이블에서 중복된 데이터 제거

정답: ②

문제 2

FULL JOIN에 대한 설명으로 옳지 않은 것은 무엇입니까?

① 두 테이블 간 모든 튜플을 결합한다.

② 두 테이블을 결합할 때 빈 칼럼은 NULL 값으로 채운다.

③ MySQL에서는 지원하지 않는다.

④ UNION ALL 연산자를 사용했을 때와 결과가 같다.

정답: ④(중복 튜플을 제거하고 합치는 UNION 연산자를 사용했을 때와 결과가 같음)

문제 **3**

다음 중 UNION 연산자를 사용할 때 주의할 점으로 바른 것은 무엇입니까?

① 두 테이블의 모든 칼럼이 동일한 값을 가져야 한다.

② 결과 집합에 중복 데이터를 포함시키지 않으려면 UNION ALL을 사용해야 한다.

③ 두 테이블의 결과 집합에 포함된 칼럼 수와 각 칼럼의 자료형이 동일해야 한다.

④ UNION 연산자는 테이블 간의 관계를 기반으로 데이터를 결합한다.

정답: ③

문제 **4**

조인을 사용하는 이유로 가장 적절한 것은 무엇입니까?

① 모든 데이터를 결합해 하나의 테이블로 만들기 위해

② 한 테이블에 있는 데이터만 필터링하기 위해

③ 데이터의 중복을 제거하기 위해

④ 여러 테이블 간의 관련 데이터를 결합해 의미 있는 정보를 추출하기 위해

정답: ④

문제 **5**

INNER JOIN의 기본 문법으로 옳은 것은 무엇입니까?

①

```
SELECT *
FROM 테이블1, 테이블2;
```

②

```
SELECT *
FROM 테이블1
INNER JOIN 테이블2;
```

③

```
SELECT *
FROM 테이블1
INNER JOIN 테이블2 ON 테이블1.칼럼 = 테이블2.칼럼;
```

④

```
SELECT *
FROM 테이블1
LEFT JOIN 테이블2;
```

정답: ③

Day 14

7장 테이블 조인하기

오늘은 별그램 DB로 조인 실습을 하고 셀프체크를 풀어봅니다.

🧐 공부할 내용(219-230p)

- 조인 실습: 별그램 DB
- 셀프체크

셀프체크의 정답을 모두 맞쳤나요? 그렇다면 추가 연습 문제를 풀며 실력을 향상해 봅시다.

1. 조인 연습 문제

다음 도서관 데이터베이스를 보고 문제에 답하세요.

```
-- members 테이블
CREATE TABLE members (
  id INTEGER AUTO_INCREMENT,     -- ID(자동 증가)
  name VARCHAR(50) NOT NULL,     -- 회원명(NULL 불가)
  email VARCHAR(100) UNIQUE,     -- 이메일(고유 값)
  phone_number CHAR(15),         -- 전화번호
  membership_status VARCHAR(20) DEFAULT 'active', -- 회원 상태(기본값: active)
  PRIMARY KEY (id)               -- 기본키 지정: id
);
-- member_profiles 테이블
CREATE TABLE member_profiles (
  id INTEGER AUTO_INCREMENT,     -- ID(자동 증가)
  date_of_birth DATE,            -- 생년월일
  address TEXT,                  -- 주소
  member_id INTEGER UNIQUE,      -- 회원_ID(고유 값)
  PRIMARY KEY (id),              -- 기본키 지정: id
  FOREIGN KEY (member_id) REFERENCES members(id) -- 외래키 지정: member_id
);
```

```sql
-- books 테이블
CREATE TABLE books (
  id INTEGER AUTO_INCREMENT,          -- ID(자동 증가)
  title VARCHAR(100) NOT NULL,        -- 도서명(NULL 불가)
  author VARCHAR(100),                -- 저자
  category VARCHAR(50),               -- 카테고리
  stock INTEGER UNSIGNED DEFAULT 0,   -- 재고(음수 불가, 기본값: 0)
  PRIMARY KEY (id)                    -- 기본키 지정: id
);
-- borrow_records 테이블
CREATE TABLE borrow_records (
  id INTEGER AUTO_INCREMENT,          -- ID(자동 증가)
  borrow_date DATE NOT NULL,          -- 대출 날짜(NULL 불가)
  return_date DATE,                   -- 반납 날짜
  member_id INTEGER NOT NULL,         -- 회원_ID
  book_id INTEGER NOT NULL,           -- 도서_ID
  PRIMARY KEY (id),                   -- 기본키 지정: id
  FOREIGN KEY (member_id) REFERENCES members(id), -- 외래키 지정: member_id
  FOREIGN KEY (book_id) REFERENCES books(id)      -- 외래키 지정: book_id
);
-- library_staff 테이블
CREATE TABLE library_staff (
  id INTEGER AUTO_INCREMENT,          -- ID(자동 증가)
  name VARCHAR(50) NOT NULL,          -- 직원명(NULL 불가)
  role VARCHAR(50) DEFAULT 'staff',   -- 역할(기본값: staff)
  employment_date DATE NOT NULL,      -- 고용 날짜(NULL 불가)
  salary INTEGER UNSIGNED CHECK (salary >= 0), -- 급여(음수 불가)
  PRIMARY KEY (id)                    -- 기본키 지정: id
);
```

문제 1

모든 회원의 이름, 생년월일, 주소를 조회하세요(회원 프로필이 없는 회원은 제외).

정답:

```sql
SELECT name, date_of_birth, address
FROM members
JOIN member_profiles ON members.id = member_profiles.member_id;
```

문제 **2**

모든 회원의 이름, 생년월일, 주소를 조회하세요(회원 프로필이 없는 경우 NULL로 출력).

정답:

```
SELECT name, date_of_birth, address
FROM members
LEFT JOIN member_profiles ON members.id = member_profiles.member_id;
```

문제 **3**

대출 기록을 보고 도서를 빌려 간 회원명과 대출한 도서명을 조회하세요.

정답:

```
SELECT name, title
FROM borrow_records
JOIN members ON borrow_records.member_id = members.id
JOIN books ON borrow_records.book_id = books.id;
```

문제 **4**

모든 도서명과 대출 날짜를 조회하세요(대출되지 않은 도서는 NULL로 출력).

정답:

```
SELECT title, borrow_date
FROM books
LEFT JOIN borrow_records ON books.id = borrow_records.book_id;
```

문제 **5**

회원과 같은 이름을 가진 직원의 이름과 역할을 조회하세요.

정답:

```
SELECT library_staff.name, role
FROM members
JOIN library_staff ON members.name = library_staff.name;
```

문제 **6**

모든 회원과 직원을 대상으로 도서관에서 주최하는 기념 행사에 초대장을 보내려고 합니다. 모든 회원과 도서관 직원의 이름을 하나의 목록으로 출력하세요(중복된 이름은 제거).

정답:

```
SELECT name
FROM members
UNION
SELECT name
FROM library_staff;
```

Day 15

8장 그룹화 분석하기

오늘은 데이터를 그룹으로 나누고 집계하는 방법인 그룹화 분석에 대해 알아보겠습니다.

🧐 공부할 내용**(231-246p)**

- 그룹화란
- 그룹화 필터링, 정렬, 조회 개수 제한

1. 그룹화란

그룹화(Grouping)는 SQL에서 데이터를 특정 컬럼의 값을 기준으로 묶어서 요약하는 방법입니다. 그룹화를 할 때는 GROUP BY 절을 사용해 기준 칼럼으로 그룹을 생성한 후 그룹별 데이터를 요약하거나 분석합니다.

```
SELECT 그룹화_칼럼, 집계_함수(일반_칼럼)
FROM 테이블명
WHERE 조건
GROUP BY 그룹화_칼럼;
```

2. 그룹화 필터링

그룹화한 결과 테이블에서 특정 조건을 만족하는 그룹의 데이터만 가져오는 것으로, GROUP BY 절로 그룹화한 대상에 HAVING 절을 추가해 수행합니다.

```
SELECT 그룹화_칼럼, 집계_함수(일반_칼럼)
FROM 테이블명
WHERE 일반_필터링_조건
GROUP BY 그룹화_칼럼
HAVING 그룹_필터링_조건;
```

예를 들어 다음과 같이 sales 테이블이 있다고 합시다.

sales

id	product	category	sales_amount	sale_date
1	Laptop	Electronics	1000	2024-01-10
2	Smartphone	Electronics	800	2024-01-12
3	Chair	Furniture	150	2024-01-13
4	Desk	Furniture	300	2024-01-14
5	Headphones	Electronics	200	2024-01-15
6	Sofa	Furniture	500	2024-01-16

여기서 카테고리별 매출 합계를 계산하는 쿼리는 다음과 같이 작성합니다.

```
SELECT category, SUM(sales_amount) AS total_sales
FROM sales
GROUP BY category;
```

결과:

category	total_sales
Electronics	2000
Furniture	950

카테고리별 매출 합계가 1000 이상인 데이터는 HAVING 절을 사용해 다음과 같이 조회합니다.

```
SELECT category, SUM(sales_amount) AS total_sales
FROM sales
GROUP BY category
HAVING total_sales >= 1000;
```

결과:

category	total_sales
Electronics	2000

3. 정렬

쿼리 결과의 데이터를 오름차순(ASC) 또는 내림차순(DESC)으로 배열하는 것으로, ORDER BY 절을 이용해 수행합니다.

```
SELECT *
FROM 테이블명
WHERE 조건
ORDER BY 정렬_칼럼1 [ASC | DESC], 정렬_칼럼2 [ASC | DESC], ...;
```

앞의 sales 테이블을 사용해 ORDER BY 절을 활용한 정렬 연습을 해 봅시다.

1. 매출액을 기준으로 오름차순 정렬하세요.

```
SELECT *
FROM sales
ORDER BY sales_amount ASC;
```

2. 매출액을 기준으로 내림차순 정렬하세요.

```
SELECT *
FROM sales
ORDER BY sales_amount DESC;
```

3. 카테고리를 기준으로 오름차순, 같은 카테고리 내에서는 매출액을 기준으로 내림차순 정렬하세요.

```
SELECT *
FROM sales
ORDER BY category ASC, sales_amount DESC;
```

4. 판매일자를 기준으로 오름차순 정렬하세요.

```
SELECT *
FROM sales
ORDER BY sale_date ASC;
```

5. 판매일자별 매출 합계를 구하고, 매출 합계를 기준으로 내림차순 정렬하세요.

```
SELECT sale_date, SUM(sales_amount) AS total_sales
FROM sales
GROUP BY sale_date
ORDER BY total_sales DESC;
```

6. 제품명을 기준으로 알파벳 내림차순 정렬하세요.

```
SELECT *
FROM sales
ORDER BY product DESC;
```

7. 카테고리별 매출 평균을 계산하고 매출 평균의 오름차순으로 정렬하세요.

```
SELECT category, AVG(sales_amount) AS avg_sales
FROM sales
GROUP BY category
ORDER BY avg_sales ASC;
```

8. 매출액이 300 이상인 행만 조회하여 매출액 기준으로 내림차순 정렬하세요.

```
SELECT *
FROM sales
WHERE sales_amount >= 300
ORDER BY sales_amount DESC;
```

4. 조회 개수 제한

상위 N개 튜플을 조회하는 것을 말합니다. LIMIT 절에 반환하려는 튜플의 개수 N을 넣어 조회합니다.

```
SELECT 칼럼1, 칼럼2, ...
FROM 테이블명
LIMIT N;
```

맨 위부터가 아니라 중간 튜플부터 조회하고 싶다면 OFFSET 키워드를 추가해 건너뛸 튜플의 개수를 지정합니다.

```
SELECT 칼럼1, 칼럼2, ...
FROM 테이블명
LIMIT N OFFSET M; -- N: 가져올 튜플의 개수, M: 건너뛸 튜플의 개수
```

sales 테이블을 사용해 LIMIT 절을 활용한 조회 개수 제한 연습을 해 봅시다.

1. 매출액이 높은 상위 3개의 항목만 조회하세요.

```
SELECT *
FROM sales
ORDER BY sales_amount DESC
LIMIT 3;
```

2. 매출액이 낮은 하위 3개의 항목만 조회하세요.

```
SELECT *
FROM sales
ORDER BY sales_amount ASC
LIMIT 3;
```

3. 매출액을 기준으로 내림차순 정렬 후 매출 상위 4위부터 6위까지 조회하세요.

```
SELECT *
FROM sales
ORDER BY sales_amount DESC
LIMIT 3 OFFSET 3;
```

5. 그룹화, 정렬, 조회 개수 제한 확인 문제

문제 1

GROUP BY 절에 대한 설명으로 옳은 것은 무엇입니까?

① 데이터를 특정 기준으로 묶어서 요약할 때 사용한다.

② 데이터를 정렬하여 출력할 때 사용한다.

③ 중복된 데이터를 제거할 때 사용한다.

④ 데이터를 필터링하여 출력할 때 사용한다.

정답: ①

문제 2

다음 쿼리의 결과로 얻을 수 있는 정보는 무엇입니까?

```
SELECT category, COUNT(*)
FROM products
GROUP BY category;
```

① 모든 제품의 목록

② 카테고리별 제품 수

③ 카테고리별 제품의 평균 가격

④ 제품별 판매 수량

정답: ②

문제 3

다음 쿼리는 무엇을 수행합니까?

```
SELECT name, price
FROM products
ORDER BY price DESC;
```

① 가격의 오름차순으로 정렬하여 제품명과 가격을 출력한다.

② 가격의 내림차순으로 정렬하여 제품명과 가격을 출력한다.

③ 제품명을 기준으로 가격을 출력한다.

④ 제품명과 가격을 정렬 없이 출력한다.

정답: ②

문제 4

ORDER BY 절을 사용할 때 기본 정렬 순서는 무엇입니까?

정답: 오름차순(ASC)

문제 5

다음 쿼리는 어떤 결과를 반환합니까?

```
SELECT name
FROM employees
LIMIT 5;
```

① 첫 5개의 이름을 반환한다.

② 마지막 5개의 이름을 반환한다.

③ 5개의 이름을 내림차순으로 정렬하여 반환한다.

④ 모든 이름을 반환한다.

정답: ①

문제 6

다음 쿼리는 무엇을 수행합니까?

```
SELECT *
FROM orders
ORDER BY order_date DESC
LIMIT 5 OFFSET 5;
```

① 가장 오래된 주문 5개를 반환한다.

② 가장 최근 주문 중 6번째부터 10번째까지를 반환한다.

③ 최근 5개의 주문을 반환한다.

④ 주문 데이터를 5개씩 나눈 첫 번째 페이지를 반환한다.

정답: ②

문제 7

HAVING 절을 사용하는 이유는 무엇입니까?

① 데이터를 정렬하기 위해

② 데이터를 결합하기 위해

③ 특정 열의 값을 필터링하기 위해

④ 그룹화된 데이터에 조건을 적용하기 위해

정답: ④

문제 8

다음 쿼리에서 WHERE 절은 언제 실행됩니까?

```
SELECT department, AVG(salary)
FROM employees
WHERE salary > 3000
GROUP BY department;
```

① 그룹화된 데이터에 적용된다.

② 데이터가 그룹화되기 전에 적용된다.

③ 데이터가 그룹화된 후 적용된다.

④ 그룹화된 데이터는 필터링할 수 없다.

정답: ②

문제 9

다음 쿼리는 무엇을 수행합니까?

```
SELECT name, sales
FROM sales_data
ORDER BY sales DESC
LIMIT 3;
```

① 가장 높은 매출을 기록한 상위 3개의 데이터를 반환한다.

② 가장 낮은 매출을 기록한 상위 3개의 데이터를 반환한다.

③ 매출이 가장 높은 데이터만 반환한다.

④ 매출 데이터를 내림차순으로 정렬하고 모든 데이터를 반환한다.

정답: ①

문제 **10**

ORDER BY 절과 GROUP BY 절의 주요 차이점은 무엇입니까?

① ORDER BY 절은 데이터를 정렬하고, GROUP BY 절은 데이터를 그룹화한다.

② ORDER BY 절은 데이터를 그룹화하고, GROUP BY 절은 데이터를 정렬한다.

③ 둘 다 데이터를 정렬하지만 방식이 다르다.

④ 둘 다 데이터를 그룹화하지만 방식이 다르다.

정답: ①

Day 16

8장 그룹화 분석하기

오늘은 마켓 DB로 그룹화 분석 실습을 하고 셀프체크를 풀어봅니다.

🧐 공부할 내용(247-264p)

- 그룹화 분석 실습: 마켓 DB
- 셀프체크

셀프체크의 정답을 모두 맞쳤나요? 그렇다면 추가 연습 문제를 풀며 실력을 향상해 봅시다.

1. 그룹화 분석 연습 문제

다음 도서관 데이터베이스를 보고 문제에 답하세요.

```
-- 회원 정보 테이블
CREATE TABLE members (
  id INTEGER AUTO_INCREMENT,        -- id
  name VARCHAR(50) NOT NULL,        -- 회원명
  join_date DATE NOT NULL,          -- 가입 날짜
  PRIMARY KEY (id)                  -- 기본키 지정: id
);
-- 도서 정보 테이블
CREATE TABLE books (
  id INTEGER AUTO_INCREMENT,        -- id
  title VARCHAR(100) NOT NULL,      -- 도서명
  author VARCHAR(50),               -- 저자
  category VARCHAR(50),             -- 카테고리
  stock INTEGER DEFAULT 0,          -- 재고
  PRIMARY KEY (id)                  -- 기본키 지정: id
);
-- 대출 기록 테이블
CREATE TABLE borrow_records (
```

```
  id INTEGER AUTO_INCREMENT,        -- id
  borrow_date DATE NOT NULL,        -- 대출 날짜
  return_date DATE,                 -- 반납 날짜
  member_id INTEGER NOT NULL,       -- 회원 id
  book_id INTEGER NOT NULL,         -- 도서 id
  PRIMARY KEY (id),                 -- 기본키 지정: id
  FOREIGN KEY (member_id) REFERENCES members(id), -- 외래 지정: member_id
  FOREIGN KEY (book_id) REFERENCES books(id)      -- 외래키 지정: book_id
);
```

문제 1

다음 쿼리의 결과로 얻을 수 있는 정보는 무엇입니까?

```
SELECT category, COUNT(*) AS total_books
FROM books
GROUP BY category;
```

① 도서별 총 재고

② 카테고리별 도서 수

③ 도서별 대출 횟수

④ 카테고리별 대출 횟수

정답: ②

문제 2

다음 쿼리의 결과로 대출량이 많은 순서로 도서를 정렬하려면 어떤 절을 추가해야 합니까?

```
SELECT title, COUNT(borrow_records.id) AS borrow_count
FROM books
JOIN borrow_records ON books.id = borrow_records.book_id
GROUP BY title;
```

① ORDER BY title;

② ORDER BY borrow_date DESC;

③ ORDER BY borrow_count DESC;

④ ORDER BY COUNT(borrow_records.id) ASC;

정답: ③

문제 3

다음 쿼리의 결과로 상위 5개 도서만 조회하려면 어떤 절을 추가해야 합니까?

```
SELECT title, COUNT(*) AS borrow_count
FROM books
JOIN borrow_records ON books.id = borrow_records.book_id
GROUP BY title
ORDER BY borrow_count DESC;
```

① LIMIT 5;

② OFFSET 5;

③ GROUP BY category;

④ HAVING borrow_count > 5;

정답: ①

문제 4

다음 쿼리의 결과로 얻을 수 있는 정보는 무엇입니까?

```
SELECT name, COUNT(borrow_records.id) AS total_borrowed
FROM members
JOIN borrow_records ON members.id = borrow_records.member_id
GROUP BY name
ORDER BY total_borrowed DESC;
```

① 도서별 총 대출량

② 도서별 대출 회원 수

③ 회원별 총 대출 금액

④ 회원별 총 대출 도서 수

정답: ④

문제 5

다음 쿼리는 무엇을 수행합니까?

```
SELECT category, SUM(stock) AS total_stock
FROM books
GROUP BY category
HAVING total_stock < 10;
```

① 총 재고가 10 이하인 도서를 조회한다.

② 총 재고가 10 이하인 카테고리를 조회한다.

③ 재고가 10 이하인 도서를 카테고리별로 그룹화한다.

④ 재고가 부족한 카테고리별 총 재고를 조회한다.

정답: ②

문제 6

다음 쿼리는 어떤 정보를 반환합니까?

```
SELECT YEAR(join_date) AS join_year, COUNT(*) AS new_members
FROM members
GROUP BY join_year
ORDER BY new_members DESC;
```

① 연도별 회원 가입 수를 오름차순으로 정렬

② 연도별 회원 가입 수를 내림차순으로 정렬

③ 연도별 총 대출 수를 내림차순으로 정렬

④ 연도별 총 대출 금액을 오름차순으로 정렬

정답: ②

Day 17

9장 서브쿼리 활용하기

오늘은 쿼리를 중첩해 작성하는 서브쿼리에 대해 알아보겠습니다

🧐 공부할 내용(265-282p)

- 서브쿼리란
- 다양한 위치에서의 서브쿼리

1. 서브쿼리란

서브쿼리란 쿼리 안에 포함된 또 다른 쿼리를 말합니다. 서브쿼리는 메인쿼리의 일부로 사용되며, 메인쿼리는 서브쿼리의 결과를 이용해 최종 실행됩니다. 서브쿼리를 이용하면 보다 복잡한 데이터 조회 및 분석을 할 수 있습니다.

```
SELECT 칼럼명1, 칼럼명2, ...
FROM 테이블명
WHERE 칼럼명 연산자 (
  서브쿼리
);
```

예를 들어 대출 기록이 있는 회원을 조회하고 싶다면 다음과 같이 서브쿼리를 작성합니다.

```
SELECT name
FROM members
WHERE id IN (
  SELECT DISTINCT member_id
  FROM borrow_records
);
```

2. 다양한 위치에서의 서브쿼리

서브쿼리는 SQL의 SELECT 절, FROM 절, JOIN 절, WHERE 절, HAVING 절 등에서 사용할 수 있습니다.

SELECT 절에서의 서브쿼리

특정 값을 계산하여 함께 출력할 때 많이 사용하며, 실행 결과로 단일 값(1×1)을 반환해야 합니다.

예: 각 회원이 대출한 도서 수를 함께 조회

```
SELECT name,
       (SELECT COUNT(*)
        FROM borrow_records
        WHERE borrow_records.member_id = members.id) AS borrowed_books
FROM members;
```

FROM 절에서의 서브쿼리

임시 테이블을 생성하여 메인쿼리에 제공합니다. 반환하는 행과 칼럼의 개수에 제한은 없습니다. 다만 결과 테이블에 반드시 별칭을 지정해야 합니다.

예: 연도별 총 대출 수를 조회

```
SELECT borrow_year, COUNT(*) AS total_borrows
FROM (
  SELECT YEAR(borrow_date) AS borrow_year
  FROM borrow_records
) AS year_borrows
GROUP BY borrow_year;
```

JOIN 절에서의 서브쿼리

서브쿼리 결과를 조인 조건으로 사용합니다. 반환하는 행과 칼럼의 개수에 제한은 없습니다. 다만 결과 테이블에는 반드시 별칭을 지정해야 합니다.

예: 가장 최근에 대출된 날짜를 각 도서의 정보와 함께 조회

```
SELECT books.title, books.author, recent_borrows.latest_borrow_date
FROM books
JOIN (
  SELECT book_id, MAX(borrow_date) AS latest_borrow_date
  FROM borrow_records
  GROUP BY book_id
) AS recent_borrows
ON books.id = recent_borrows.book_id;
```

WHERE 절에서의 서브쿼리

서브쿼리 결과를 필터링 조건으로 사용합니다. 단일 값(1×1) 또는 다중 행의 단일 칼럼(N×1)을 반환할 수 있습니다. 만약 단일 값을 반환할 경우 메인쿼리에서는 비교 연산자(=, !=, >, <, >=, <=)를 사용하고, 다중 행의 단일 칼럼을 반환할 경우 메인쿼리에서는 IN, ANY, ALL, EXISTS 연산자를 사용합니다.

예: 대출 기록이 있는 도서를 조회

```
SELECT title, author
FROM books
WHERE id IN (
  SELECT DISTINCT book_id
  FROM borrow_records
);
```

HAVING 절에서의 서브쿼리

그룹화된 데이터의 집계 결과를 비교하는 조건으로 사용합니다. 그룹화 필터링을 수행하기 위한 것이므로 WHERE 절의 서브쿼리와 같이 단일 값(1×1) 또는 다중 행의 단일 칼럼(N×1)을 반환할 수 있습니다.

예: 회원별 대출 도서 수가 전체 평균 대출 도서 수 이상인 회원 조회

```
SELECT member_id, COUNT(*) AS total_borrows
FROM borrow_records
GROUP BY member_id
HAVING COUNT(*) >= (
  SELECT AVG(total_borrowed)
  FROM (
    SELECT COUNT(*) AS total_borrowed
    FROM borrow_records
    GROUP BY member_id
  ) AS member_borrows
);
```

3. 서브쿼리 확인 문제

문제 1

서브쿼리에 대한 설명으로 옳은 것은 무엇입니까?

① 하나의 SQL 문에서 다른 SQL 문을 중첩하여 사용하는 것

② 여러 테이블을 결합하여 하나의 결과를 반환하는 것

③ 특정 조건을 기준으로 데이터를 필터링하는 명령

④ 쿼리 실행 결과를 정렬하는 방식

정답: ①

문제 2

다음 중 서브쿼리를 사용할 수 없는 절은 무엇입니까?

① SELECT 절 ② WHERE 절

③ JOIN 절 ④ LIMIT 절

정답: ④

문제 3

서브쿼리와 JOIN의 차이에 대한 설명으로 옳은 것은 무엇입니까?

① 서브쿼리는 데이터를 결합하고, JOIN은 데이터를 필터링한다.

② 서브쿼리는 독립적으로 실행되며, JOIN은 두 테이블 간의 관계를 결합한다.

③ JOIN은 두 테이블 간의 모든 데이터를 결합하고, 서브쿼리는 특정 데이터를 필터링한다.

④ JOIN은 임시 테이블을 생성하고, 서브쿼리는 모든 데이터를 반환한다.

정답: ②

문제 4

다음 쿼리에서 서브쿼리가 반환하는 데이터는 무엇입니까?

```
SELECT name
FROM members
WHERE id IN (
  SELECT member_id
  FROM borrow_records
);
```

① 회원의 이름을 포함한 모든 정보

② 대출 기록에서 조회된 회원의 이름

③ 대출 기록에서 조회된 회원 ID

④ 대출 기록에서 조회된 도서 ID

정답: ③

문제 **5**

다음 중 서브쿼리가 특정 절에서 사용되는 이유와 그 역할로 가장 적절한 것은 무엇입니까?

① SELECT 절: 서브쿼리의 결과를 메인쿼리의 출력 값으로 계산하여 제공한다.

② FROM 절: 서브쿼리의 결과를 임시 테이블처럼 활용하여 메인쿼리에서 참조한다.

③ WHERE 절: 서브쿼리의 결과를 조건으로 사용하여 메인쿼리의 데이터를 필터링한다.

④ HAVING 절: 그룹화된 데이터의 집계 결과를 조건으로 제한한다.

⑤ 모두 맞다.

정답: ⑤

Day 18

9장 서브쿼리 활용하기

오늘은 서브쿼리와 함께 사용되는 IN, ANY, ALL, EXISTS 연산자에 대해 알아보고 셀프체크를 풀어보겠습니다.

🧐 공부할 내용(283-298p)

- IN, ANY, ALL, EXISTS
- 셀프체크

셀프체크의 정답을 모두 맞쳤나요? 그렇다면 추가 연습 문제를 풀며 실력을 향상해 봅시다.

1. 서브쿼리 연습 문제

회사 데이터베이스를 보고 문제에 답하세요.

```
-- 부서 테이블
CREATE TABLE departments (
  id INTEGER AUTO_INCREMENT,        -- id
  name VARCHAR(50) NOT NULL,        -- 부서명
  location VARCHAR(50),             -- 위치
  PRIMARY KEY (id)                  -- 기본키 지정:id
);
-- 직원 테이블
CREATE TABLE employees (
  id INTEGER AUTO_INCREMENT,        -- id
  name VARCHAR(50) NOT NULL,        -- 직원명
  hire_date DATE NOT NULL,          -- 입사 날짜
  salary INTEGER NOT NULL,          -- 급여
  department_id INTEGER,            -- 부서 id
  PRIMARY KEY (id),                 -- 기본키 지정: id
  FOREIGN KEY (department_id) REFERENCES departments(id) -- 외래키 지정:
department_id
```

```
);
-- 프로젝트 테이블
CREATE TABLE projects (
  id INTEGER AUTO_INCREMENT,        -- id
  name VARCHAR(100) NOT NULL,       -- 프로젝트명
  start_date DATE NOT NULL,         -- 시작 날짜
  end_date DATE,                    -- 종료 날짜
  PRIMARY KEY (id)                  -- 기본키 지정: id
);
-- 직원-프로젝트 테이블 (다대다 관계)
CREATE TABLE employee_projects (
  id INTEGER AUTO_INCREMENT,        -- id
  employee_id INTEGER NOT NULL,     -- 직원 id
  project_id INTEGER NOT NULL,      -- 프로젝트 id
  PRIMARY KEY (id),                 -- 기본키 지정: id
  FOREIGN KEY (employee_id) REFERENCES employees(id), -- 외래키 지정:
employee_id
  FOREIGN KEY (project_id) REFERENCES projects(id)    -- 외래키 지정:
project_id
);
-- 급여 기록 테이블
CREATE TABLE salary_records (
  id INTEGER AUTO_INCREMENT,        -- id
  salary_date DATE NOT NULL,        -- 급여 지급 날짜
  amount INTEGER NOT NULL,          -- 지급 금액
  employee_id INTEGER NOT NULL,     -- 직원 id
  PRIMARY KEY (id),                 -- 기본키 지정: id
  FOREIGN KEY (employee_id) REFERENCES employees(id) -- 외래키 지정:
employee_id
);
```

문제 1: SELECT 절에서의 서브쿼리

각 직원의 이름과 참여 중인 프로젝트 수를 조회하세요.

정답:

```
SELECT name,
       (SELECT COUNT(*)
        FROM employee_projects
        WHERE employee_projects.employee_id = employees.id) AS
project_count
FROM employees;
```

문제 **2: WHERE** 절에서의 서브쿼리

특정 부서(예: IT 부서)의 직원 이름을 조회하세요.

정답:

```
SELECT name
FROM employees
WHERE department_id = (
  SELECT id
  FROM departments
  WHERE name = 'IT'
);
```

문제 **3: FROM** 절에서의 서브쿼리

부서별 직원 수를 조회하세요.

정답:

```
SELECT department_name, COUNT(*) AS employee_count
FROM (
  SELECT d.name AS department_name, e.id AS employee_id
  FROM employees e
  JOIN departments d ON e.department_id = d.id
) AS department_employees
GROUP BY department_name;
```

문제 **4: JOIN** 절에서의 서브쿼리

가장 높은 급여를 받은 직원의 이름과 급여를 조회하세요.

정답:

```
SELECT employees.name, salary_records.amount
FROM employees
JOIN salary_records ON employees.id = salary_records.employee_id
WHERE salary_records.amount = (
  SELECT MAX(amount)
  FROM salary_records
);
```

문제 **5: HAVING** 절에서의 서브쿼리

부서별 평균 급여가 전체 평균 급여 이상인 부서명을 조회하세요.

정답:

```
SELECT departments.name AS department_name, AVG(salary_records.amount)
AS avg_department_salary
FROM employees
JOIN salary_records ON employees.id = salary_records.employee_id
JOIN departments ON employees.department_id = departments.id
GROUP BY departments.name
HAVING avg_department_salary >= (
  SELECT AVG(amount)
  FROM salary_records
);
```

문제 **6:** 복합 조건을 조합한 서브쿼리

가장 많은 직원이 참여한 프로젝트명을 조회하세요.

정답:

```
SELECT name
FROM projects
WHERE id = (
  SELECT project_id
  FROM employee_projects
  GROUP BY project_id
  ORDER BY COUNT(*) DESC
  LIMIT 1
);
```

Day 19

10장 데이터 모델링과 정규화

데이터 모델링이란 현실 세계의 어떤 사물이나 현상을 추상화(모형화)해 데이터 간 구조와 관계를 정의한 '데이터 모델'을 만드는 과정입니다. 오늘은 데이터 모델링의 개념과 구성 요소를 알아보고, 쇼핑몰을 데이터 모델링하는 과정을 실습해 보겠습니다

🧐 공부할 내용**(299-327p)**

- 데이터 모델링이란
- 데이터 모델의 구성 요소
- 데이터 모델링 실습: 쇼핑몰 DB

1. 데이터 모델링이란

데이터 모델링(Data Modeling)은 데이터를 효율적으로 저장하고 관리하기 위해 데이터 구조를 설계하는 과정입니다. 데이터베이스의 논리적, 물리적 구조를 명확히 정의하여 데이터의 무결성과 효율적인 데이터 처리를 보장하기 위해 데이터 모델링을 수행합니다.

데이터 모델링의 목적

- 데이터 중복 방지
- 데이터 무결성 유지
- 비즈니스 요구사항 반영
- 데이터 처리 성능 최적화

데이터 모델링 단계

① 개념적 데이터 모델링

데이터베이스를 구축하기 전, 비즈니스 요구사항을 파악하여 데이터를 추상화하는 단계입니다. 산출물로 ER 다이어그램(Entity-Relationship Diagram)이 나오며, 이를 통해 비즈니스에 필요한 주요 엔티티(Entity)와 속성(Attribute)을 정의할 수 있습니다.

② 논리적 데이터 모델링

개념적 데이터 모델을 데이터베이스 구조로 논리적으로 정의하는 단계입니다. 산출물로 논리적 데이터 모델(테이블, 칼럼, 키 정의)을 얻습니다. 이 단계에서는 각 엔티티를 테이블로, 각 속성을 칼럼으로 정의한 후 각 칼럼의 자료형과 제약 조건을 설정합니다.

③ 물리적 데이터 모델링

논리적 데이터 모델을 실제 DBMS에 구현 가능한 형태로 변환하는 단계입니다. 산출물로 물리적 데이터베이스 스키마(SQL 생성 스크립트)가 생성됩니다. 이 단계에서는 데이터베이스 인덱스 설계, 데이터베이스 파티셔닝 또는 분산 설계, 스토리지 최적화 및 성능 튜닝 등을 수행합니다.

데이터 모델링의 주요 용어

- 엔티티: 데이터베이스에서 관리해야 할 데이터의 단위를 말하며(예: 학생, 도서, 주문 등), 논리적 데이터 모델링을 거쳐 '테이블'이 됩니다.

- 속성: 엔티티가 가지는 특징이나 정보를 나타내는 항목을 말하며(예: 학생 엔티티의 이름, 나이, 학번 등), 논리적 데이터 모델링을 거쳐 '칼럼'이 됩니다.

- 관계: 2개 이상의 엔티티 간의 연관성을 말하며(예: 학생과 수업 간의 '수강' 관계), '중간 테이블'로 구현됩니다.

- 기본키: 각 엔티티를 고유하게 식별하는 속성입니다(예: 학생의 학번, 주문의 주문 ID).

- 외래키: 다른 테이블의 기본키를 참조하여 관계를 정의하는 속성입니다(예: 주문 테이블의 고객 ID는 고객 테이블블의 기본키를 참조하는 외래키).

데이터 모델링 실습

다음 시나리오를 분석해 데이터 모델을 수행해 봅시다.

- 학생(Student)이 수업(Course)을 수강한다(Enrollment).

① 개념적 모델링(시나리오 분석해 엔티티와 관계 정의)

- 엔티티 정의: student, course, enrollment
- 관계 정의: 학생과 수업은 다대다 관계

② 논리적 모델링(테이블 정의)

- students(id, name, age)
- courses(id, title, credits)
- enrollment(grade, student_id, course_id)

③ 물리적 모델링(실제 DBMS에 맞는 SQL로 구현)

```
-- students 테이블
  CREATE TABLE students (
    id INTEGER,
    name VARCHAR(50) NOT NULL,
    age INTEGER CHECK (age > 0),
    PRIMARY KEY (id)
  );
-- courses 테이블
  CREATE TABLE courses (
    id INTEGER,
    title VARCHAR(100) NOT NULL,
    credits INTEGER NOT NULL,
    PRIMARY KEY (id)
  );
--  enrollment 테이블
  CREATE TABLE enrollment (
    id INTEGER,
    grade CHAR(1),
    student_id INTEGER NOT NULL,
    course_id INTEGER NOT NULL,
    PRIMARY KEY (id),
    FOREIGN KEY (student_id) REFERENCES students(id),
    FOREIGN KEY (course_id) REFERENCES courses(id)
  );
```

2. 데이터 모델링 확인 문제

문제 1

데이터 모델링에 대한 설명으로 옳은 것은 무엇입니까?

① 데이터를 효율적으로 저장하고 관리하기 위해 데이터 구조를 설계하는 과정이다.

② 데이터를 시각적으로 분석하기 위한 차트를 생성하는 과정이다.

③ 데이터베이스의 성능을 튜닝하는 과정을 의미한다.

④ 데이터베이스의 백업과 복원을 설계하는 과정이다.

정답: ①

문제 2

데이터 모델링의 주요 목적에 해당하지 않는 것은 무엇입니까?

① 데이터 중복 방지

② 데이터 무결성 유지

③ 비즈니스 요구사항 반영

④ 데이터 시각화 향상

정답: ④

문제 3

다음 중 데이터 모델링의 단계가 올바르게 나열된 것은 무엇입니까?

① 논리적 모델링 → 물리적 모델링 → 개념적 모델링

② 물리적 모델링 → 논리적 모델링 → 개념적 모델링

③ 개념적 모델링 → 논리적 모델링 → 물리적 모델링

④ 논리적 모델링 → 개념적 모델링 → 물리적 모델링

정답: ③

문제 **4**

다음 중 엔티티에 대한 설명으로 옳은 것은 무엇입니까?

① 데이터베이스의 기본키로만 사용되는 속성이다.

② 데이터의 속성이나 특징을 나타내는 요소이다.

③ 데이터베이스에서 관리해야 할 데이터 단위이다.

④ 데이터베이스 간의 관계를 나타내는 요소이다.

정답: ③

문제 **5**

다음 중 속성에 해당하는 것은 무엇입니까?

① 회원 테이블의 'id'와 '회원명'

② 회원과 주문 간의 관계

③ 회원 테이블과 주문 테이블

④ 주문 테이블의 '고객 id'와 '상품 id'

정답: ①

문제 **6**

기본키와 외래키에 대한 설명으로 옳은 것은 무엇입니까?

① 기본키는 데이터를 고유하게 식별하며, 외래키는 다른 테이블과의 관계를 나타낸다.

② 기본키는 두 테이블 간의 관계를 나타내며, 외래키는 데이터를 고유하게 식별한다.

③ 기본키와 외래키는 동일한 역할을 한다.

④ 외래키는 모든 테이블에 필수적으로 존재해야 한다.

정답: ①

문제 7

개념적 데이터 모델링의 결과로 생성되는 산출물은 무엇입니까?

① SQL 스크립트 ② 테이블 정의서

③ ER 다이어그램 ④ 데이터베이스 실행 로그

정답: ③

문제 8

다음 중 관계에 대한 설명으로 옳은 것은 무엇입니까?

① 2개 이상의 테이블 간의 데이터 중복 저장되는 것을 방지한다.

② 개체 간의 연관성을 정의한다.

③ 데이터베이스의 성능을 최적화한다.

④ 테이블의 속성을 나열한다.

정답: ②

문제 9

데이터 무결성을 유지하기 위해 설정하는 제약 조건이 아닌 것은 무엇입니까?

① NOT NULL ② UNIQUE

③ AUTO_INCREMENT ④ FOREIGN KEY

정답: ④

Day 20

10장 데이터 모델링과 정규화

드디어 마지막 날입니다. 오늘은 데이터 모델링이 잘 됐는지 검증하고 개선하는 정규화에 대해 알아보고 데이터 모델링에 관한 셀프체크를 풀어보겠습니다.

🧐 공부할 내용(328-338p)

- 정규화
- 셀프체크

셀프체크의 정답을 모두 맞쳤나요? 그렇다면 추가 연습 문제를 풀며 실력을 향상해 봅시다.

1. 데이터 모델링과 정규화 연습 문제

문제 1

도서관 데이터베이스를 설계할 때, 다음 중 개념적 모델링 단계에서 수행해야 하는 작업은 무엇입니까?

① 회원, 도서, 대출 기록 등의 개체와 속성을 정의한다.

② 도서 테이블의 재고 칼럼에 UNSIGNED 제약 조건을 추가한다.

③ 대출 기록 테이블의 외래키 관계를 명시한다.

④ 테이블의 인덱스 설계를 통해 쿼리 성능을 최적화한다.

정답: ①

문제 **2**

도서관 데이터베이스의 회원과 대출 기록 간의 관계로 가장 적합한 것은 무엇입니까?

① 일대일 관계 ② 일대다 관계

③ 다대다 관계 ④ 다대일 관계

정답: ②(한 명의 회원은 여러 대출 기록을 가질 수 있지만, 각 대출 기록은 특정 회원에만 속하므로 일대다 관계가 성립)

문제 **3**

도서관 데이터베이스의 물리적 모델링 단계에서 수행해야 하는 작업은 무엇입니까?

① 대출 기록 테이블에서 borrow_date 칼럼에 기본값으로 현재 날짜를 설정한다.

② 회원과 대출 기록 간의 관계를 일대다로 정의한다.

③ 회원 테이블과 대출 기록 테이블 간의 ER 다이어그램을 작성한다.

④ 대출 기록 테이블에서 borrow_date 칼럼을 생성한다.

정답: ①

문제 **4**

제1정규형, 제2정규형, 제3정규형으로 변환하는 과정을 바르게 나열한 것은 무엇입니까?

1. 기본키를 설정하고 모든 속성이 원자값을 가지도록 정리한다.
2. 복합키에서 부분 함수 종속을 제거한다.
3. 이행적 함수 종속을 제거한다.

① 1 → 2 → 3 ② 1 → 3 → 2

③ 2 → 1 → 3 ④ 3 → 2 → 1

정답: ①

문제 **5**

다음 테이블은 정규화가 제대로 이루어지지 않았습니다. 위반된 가장 낮은 정규형은 무엇입니까?

member_id	book_id	borrow_date	member_name	book_title	category
1	101	2024-01-01	홍길동	자바의 정석	프로그래밍
2	102	2024-01-02	이순신	파이썬 마스터	프로그래밍
1	103	2024-01-03	홍길동	SQL 마스터	데이터베이스
3	101	2024-01-04	김영희	자바의 정석	프로그래밍

① 제1정규형

② 제2정규형

③ 제3정규형

④ 없음

정답: ②

해설:

- 각 속성은 원자값(더 이상 분해할 수 없는 값)을 가지고, 같은 속성끼리는 모두 동일한 자료형을 가지며, 각 인스턴스는 고유하므로 제1정규형을 만족한다.
- 기본키 후보는 {member_id, book_id}의 복합키이며, 이때 member_name은 member_id에만 종속되고, book_title과 category는 book_id에만 종속된다. 따라서 이 경우 부분 함수 종속이 존재하므로, 제2정규형에 위반한다.