SQL의 데이터 조작 기능

데이터의 검색 · 삽입 · 수정 · 삭제

그림 7-5 SQL의 데이터 조작 기능

실습 준비 판매 데이터베이스 세팅

아래 SQL을 전부 복사해서 실행하세요 (맨 처음 한번만!)

```
DROP TABLE IF EXISTS 주문; DROP TABLE IF EXISTS 제품; DROP TABLE IF EXISTS 고객;
CREATE TABLE 고객 (고객아이디 VARCHAR(20) PRIMARY KEY, 고객이름 VARCHAR(10),
  나이 INT, 등급 VARCHAR(10), 직업 VARCHAR(20), 적립금 INT DEFAULT 0);
CREATE TABLE 제품 (제품번호 CHAR(3) PRIMARY KEY, 제품명 VARCHAR(20),
  재고량 INT, 단가 INT, 제조업체 VARCHAR(20));
CREATE TABLE 주문 (주문번호 CHAR(3) PRIMARY KEY, 주문고객 VARCHAR(20),
  주문제품 CHAR(3), 수량 INT, 배송지 VARCHAR(30), 주문일자 DATE,
  FOREIGN KEY(주문고객) REFERENCES 고객(고객아이디),
  FOREIGN KEY(주문제품) REFERENCES 제품(제품번호));
INSERT INTO 고객 VALUES ('apple','정소화',20,'gold','학생',1000),
  ('banana','김선우',25,'vip','간호사',2500),('carrot','고명석',28,'gold','교사',4500),
  ('orange','김용욱',22,'silver','학생',0),('melon','성원용',35,'gold','회사원',5000),
  ('peach','오형준',NULL,'silver','의사',300),('pear','채광주',31,'silver','회사원',500);
INSERT INTO 제품 VALUES ('p01','그냥만두',5000,4500,'대한식품'),
  ('p02','매운쫄면',2500,5500,'민국푸드'),('p03','쿵떡파이',3600,2600,'한빛제과'),
  ('p04','맛난초콜릿',1250,500,'한빛제과'),('p05','얼큰라면',2200,1200,'대한식품'),
  ('p06','통통우동',1000,1550,'민국푸드'),('p07','달콤비스킷',1650,1500,'한빛제과');
INSERT INTO 주문 VALUES ('o01','apple','p03',10,'서울시','2022-01-01'),
  ('o02','melon','p01',5,'인천시','2022-01-10'),('o03','banana','p06',45,'경기도','2022-01-11'),
  ('o04','carrot','p02',8,'부산시','2022-02-01'),('o05','apple','p06',36,'경기도','2022-02-20'),
  ('o06','orange','p02',19,'충청도','2022-03-02'),('o07','apple','p03',22,'서울시','2022-03-15'),
  ('o08','pear','p02',50,'강원도','2022-04-10'),('o09','banana','p04',15,'전라도','2022-04-11'),
  ('o10','carrot','p03',20,'경기도','2022-05-22');
-- ▶ 확인:
SELECT * FROM 고객;  SELECT * FROM 제품;  SELECT * FROM 주문;
```

데이터의 검색

SELECT 문 기본 형식

고객 테이블

고객아이디	고객이름	나이	등급	직업	적립금
apple	정소화	20	gold	학생	1000
banana	김선우	25	vip	간호사	2500
carrot	고명석	28	gold	교사	4500
orange	김용욱	22	silver	학생	0
melon	성원용	35	gold	회사원	5000
peach	오형준	NULL	silver	의사	300
pear	채광주	31	silver	회사원	500

제품 테이블

제품번호	제품명	재고량	단가	제조업체
p01	그냥만두	5000	4500	대한식품
p02	매운쫄면	2500	5500	민국푸드
p03	쿵떡파이	3600	2600	한빛제과
p04	맛난초콜릿	1250	2500	한빛제과
p05	얼큰라면	2200	1200	대한식품
p06	통통우동	1000	1550	민국푸드
p07	달콤비스킷	1650	1500	한빛제과

주문 테이블

주문번호	주문고객	주문제품	수량	배송지	주문일자
o01	apple	p03	10	서울시 마포구	2022-01-01
o02	melon	p01	5	인천시 계양구	2022-01-10
o03	banana	p06	45	경기도 부천시	2022-01-11
o04	carrot	p02	8	부산시 금정구	2022-02-01
o05	melon	p06	36	경기도 용인시	2022-02-20
o06	banana	p01	19	충청북도 보은군	2022-03-02
o07	apple	p03	22	서울시 영등포구	2022-03-15
o08	pear	p02	50	강원도 춘천시	2022-04-10
o09	banana	p04	15	전라남도 목포시	2022-04-11
o10	carrot	p03	20	경기도 안양시	2022-05-22

그림 7-6 데이터 조작 예제에서 사용하는 판매 데이터베이스 : 고객, 제품, 주문 테이블

데이터의 검색

기본 검색

```
SELECT  [ ALL | DISTINCT ] 속성_리스트
FROM    테이블_리스트;
```

예제 7-10

고객 테이블에서 고객아이디, 고객이름, 등급 속성을 검색해보자.

```
▶▶ SELECT    고객아이디, 고객이름, 등급
   FROM      고객;
```

결과 테이블

	고객아이디	고객이름	등급
1	apple	정소화	gold
2	banana	김선우	vip
3	carrot	고명석	gold
4	orange	김용욱	silver
5	melon	성원용	gold
6	peach	오형준	silver
7	pear	채광주	silver

데이터의 검색

기본 검색

```
SELECT  [ ALL ¦ DISTINCT ] 속성_리스트
FROM    테이블_리스트;
```

예제 7-11

고객 테이블에 존재하는 모든 속성을 검색해보자.

```
▶▶ SELECT   고객아이디, 고객이름, 나이, 등급, 직업, 적립금
   FROM     고객;
```

결과 테이블

	고객아이디	고객이름	나이	등급	직업	적립금
1	apple	정소화	20	gold	학생	1000
2	banana	김선우	25	vip	간호사	2500
3	carrot	고명석	28	gold	교사	4500
4	orange	김용욱	22	silver	학생	0
5	melon	성원용	35	gold	회사원	5000
6	peach	오형준	(null)	silver	의사	300
7	pear	채광주	31	silver	회사원	500

데이터의 검색

기본 검색

```
SELECT  [ ALL ¦ DISTINCT ] 속성_리스트
FROM    테이블_리스트;
```

예제 7-12

고객 테이블에 존재하는 모든 속성을 검색해보자.

```
SELECT    *
FROM      고객;
```

결과 테이블

	고객아이디	고객이름	나이	등급	직업	적립금
1	apple	정소화	20	gold	학생	1000
2	banana	김선우	25	vip	간호사	2500
3	carrot	고명석	28	gold	교사	4500
4	orange	김용욱	22	silver	학생	0
5	melon	성원용	35	gold	회사원	5000
6	peach	오형준	(null)	silver	의사	300
7	pear	채광주	31	silver	회사원	500

데이터의 검색

기본 검색

```
SELECT  [ ALL ¦ DISTINCT ] 속성_리스트
FROM    테이블_리스트;
```

예제 7-13

제품 테이블에서 제조업체를 검색해보자.

```
SELECT    제조업체
FROM      제품;
```

결과 테이블

	제조업체
1	대한식품
2	민국푸드
3	한빛제과
4	한빛제과
5	대한식품
6	민국푸드
7	한빛제과

데이터의 검색

기본 검색

```
SELECT  [ ALL ¦ DISTINCT ] 속성_리스트
FROM    테이블_리스트;
```

예제 7-14

제품 테이블에서 제조업체를 검색하되, ALL 키워드를 사용해보자.

```
SELECT    ALL    제조업체
FROM      제품;
```

결과 테이블

	제조업체
1	대한식품
2	민국푸드
3	한빛제과
4	한빛제과
5	대한식품
6	민국푸드
7	한빛제과

데이터의 검색

기본 검색

```
SELECT  [ ALL ¦ DISTINCT ] 속성_리스트
FROM    테이블_리스트;
```

예제 7-15

제품 테이블에서 제조업체 속성을 중복 없이 검색해보자.

```
▶▶ SELECT    DISTINCT    제조업체
   FROM      제품;
```

결과 테이블

	제조업체
1	대한식품
2	민국푸드
3	한빛제과

실습 ① SELECT 기본 + DISTINCT

STEP 1 전체 조회 — SELECT *

```
SELECT * FROM 고객;
```

STEP 2 특정 속성만 — 프로젝트(π)

```
SELECT 고객아이디, 고객이름, 등급 FROM 고객;
```

STEP 3 중복 포함 vs 중복 제거 — 행 수 비교!

```
SELECT ALL 제조업체 FROM 제품;     -- 7행 (중복 포함)
SELECT DISTINCT 제조업체 FROM 제품; -- 3행 (중복 제거)
```

STEP 4 AS 별칭 + 산술식

```
SELECT 제품명, 단가 AS 현재가, 단가+500 AS 인상가 FROM 제품;
```

데이터의 검색

기본 검색 — AS (컬럼 Alias)

AS: 컬럼 Alias

SELECT 문으로 조회된 집합에서 '단가'의 속성 이름을 '가격'으로 변경함

예제 7-16

제품 테이블에서 제품명과 단가를 검색하되, 단가를 가격이라는 새 이름으로 출력해보자.

```
▶▶ SELECT   제품명, 단가 AS 가격
   FROM     제품;
```

결과 테이블

	제품명	가격
1	그냥만두	4500
2	매운쫄면	5500
3	쿵떡파이	2600
4	맛난초콜릿	2500
5	얼큰라면	1200
6	통통우동	1550
7	달콤비스킷	1500

데이터의 검색

산술식을 이용한 검색

산술식 검색
속성값에 대해 사칙연산(+, -, *, /)을 수행해서 쿼리할 수 있음

예제 7-17

제품 테이블에서 제품명과 단가 속성을 검색하되, 단가에 500원을 더해 '조정 단가'라는 새 이름으로 출력해보자.

```
▶▶ SELECT   제품명, 단가 + 500 AS "조정 단가"
   FROM     제품;
```

결과 테이블

	제품명	조정 단가
1	그냥만두	5000
2	매운쫄면	6000
3	쿵떡파이	3100
4	맛난초콜릿	3000
5	얼큰라면	1700
6	통통우동	2050
7	달콤비스킷	2000

데이터의 검색

조건 검색

```
SELECT   [ ALL ¦ DISTINCT ] 속성_리스트
FROM     테이블_리스트
[ WHERE  조건 ];
```

표 7-3 논리 연산자

연산자	의미
AND	모든 조건을 만족해야 검색한다.
OR	여러 조건 중 한 가지만 만족해도 검색한다.
NOT	조건을 만족하지 않는 것만 검색한다.

표 7-2 비교 연산자

연산자	의미
=	같다.
< >	다르다.
<	작다.
>	크다.
<=	작거나 같다.
>=	크거나 같다.

데이터의 검색

조건 검색 (계속)

```
SELECT   [ ALL ¦ DISTINCT ] 속성_리스트
FROM     테이블_리스트
[ WHERE  조건 ];
```

예제 7-18

제품 테이블에서 한빛제과가 제조한 제품의 제품명, 재고량, 단가를 검색해보자.

```
▶▶ SELECT     제품명, 재고량, 단가
   FROM       제품
   WHERE      제조업체 = '한빛제과';
```

결과 테이블

	제품명	재고량	단가
1	쿵떡파이	3600	2600
2	맛난초콜릿	1250	2500
3	달콤비스킷	1650	1500

데이터의 검색

조건 검색 (계속)

```
SELECT   [ ALL ¦ DISTINCT ] 속성_리스트
FROM     테이블_리스트
[ WHERE  조건 ];
```

예제 7-19

주문 테이블에서 apple 고객이 15개 이상 주문한 주문제품, 수량, 주문일자를 검색해보자.

```
SELECT     주문제품, 수량, 주문일자
FROM       주문
WHERE      주문고객 = 'apple'  AND  수량 >= 15;
```

결과 테이블

	주문제품	수량	주문일자
1	p03	22	22/03/15

데이터의 검색

조건 검색 (계속)

예제 7-20

주문 테이블에서 apple 고객이 주문했거나 15개 이상 주문된 제품의 주문제품, 수량, 주문일자, 주문고객을 검색해보자.

```
SELECT   주문제품, 수량, 주문일자, 주문고객
FROM     주문
WHERE    주문고객 = 'apple'  OR  수량 >= 15;
```

결과 테이블

	주문제품	수량	주문일자	주문고객
1	p03	10	22/01/01	apple
2	p06	45	22/01/11	banana
3	p06	36	22/02/20	melon
4	p01	19	22/03/02	banana
5	p03	22	22/03/15	apple
6	p02	50	22/04/10	pear
7	p04	15	22/04/11	banana
8	p03	20	22/05/22	carrot

데이터의 검색

조건 검색 (계속)

예제 7-21

제품 테이블에서 단가가 2,000원 이상이면서 3,000원 이하인 제품의 제품명, 단가, 제조업체를 검색해보자.

```
▶▶ SELECT   제품명, 단가, 제조업체
   FROM     제품
   WHERE    단가 >= 2000  AND  단가 <= 3000;
```

결과 테이블

	제품명	단가	제조업체
1	쿵떡파이	2600	한빛제과
2	맛난초콜릿	2500	한빛제과

NOTE SQL 문을 작성할 때 천 단위를 구분하는 콤마(,)는 숫자 값에 사용하지 않는다.

실습 ② WHERE 조건 검색

STEP 1 단일 조건 — 비교 연산자

```
SELECT * FROM 제품 WHERE 제조업체 = '한빛제과';
SELECT * FROM 제품 WHERE 단가 >= 2000;
```

STEP 2 AND — 두 조건 모두 만족

```
SELECT * FROM 제품
  WHERE 제조업체 = '한빛제과' AND 재고량 >= 3000;
```

STEP 3 OR — 하나만 만족해도 OK

```
SELECT * FROM 고객
  WHERE 등급 = 'gold' OR 등급 = 'vip';
```

STEP 4 BETWEEN, IN — 범위/집합 검색

```
SELECT * FROM 제품 WHERE 단가 BETWEEN 1000 AND 2000;
SELECT * FROM 주문 WHERE 주문고객 IN ('apple','banana');
```

데이터의 검색

LIKE를 이용한 검색

% > _ (퍼센트가 밑줄보다 범위가 넓음)

표 7-4 LIKE 키워드와 함께 사용할 수 있는 기호

기호	설명
%	0개 이상의 문자 (문자의 내용과 개수는 상관 없음)
_	1개의 문자 (문자의 내용은 상관 없음)

표 7-5 LIKE 키워드의 사용 예

사용 예	설명
LIKE '데이터%'	데이터로 시작하는 문자열 (데이터로 시작하기만 하면 길이는 상관 없음)
LIKE '%데이터'	데이터로 끝나는 문자열 (데이터로 끝나기만 하면 길이는 상관 없음)
LIKE '%데이터%'	데이터가 포함된 문자열
LIKE '데이터 _ _ _'	데이터로 시작하는 6자 길이의 문자열
LIKE '_ _ 한%'	세 번째 글자가 '한'인 문자열

데이터의 검색

LIKE를 이용한 검색 (예제)

예제 7-22

고객 테이블에서 성이 김 씨인 고객의 고객이름, 나이, 등급, 적립금을 검색해보자.

```
SELECT    고객이름, 나이, 등급, 적립금
FROM      고객
WHERE     고객이름 LIKE '김%';
```

결과 테이블

	고객이름	나이	등급	적립금
1	김선우	25	vip	2500
2	김용욱	22	silver	0

데이터의 검색

LIKE를 이용한 검색 (예제)

예제 7-23

고객 테이블에서 고객아이디가 5자인 고객의 고객아이디, 고객이름, 등급을 검색해보자.

```
SELECT    고객아이디, 고객이름, 등급
FROM      고객
WHERE     고객아이디 LIKE '_ _ _ _ _';
```

결과 테이블

	고객아이디	고객이름	등급
1	apple	정소화	gold
2	melon	성원용	gold
3	peach	오형준	silver

실습 ③ LIKE 패턴 매칭

STEP 1 % — 0개 이상의 임의 문자

```
SELECT * FROM 고객 WHERE 고객이름 LIKE '김%';
-- 김선우, 김용욱 → 2행
```

STEP 2 %의 위치별 차이 비교

```
SELECT * FROM 제품 WHERE 제품명 LIKE '%만두';
SELECT * FROM 제품 WHERE 제품명 LIKE '%콩%';
```

STEP 3 _ — 정확히 1글자

```
SELECT * FROM 고객 WHERE 고객이름 LIKE '_소화';
-- 3글자 중 뒤 2글자가 '소화' → 정소화
```

STEP 4 NOT LIKE

```
SELECT * FROM 고객 WHERE 고객이름 NOT LIKE '김%';
-- 김으로 시작 안 하는 고객 → 5명
```

데이터의 검색

NULL을 이용한 검색

IS NULL / IS NOT NULL

예제 7-24

고객 테이블에서 나이가 아직 입력되지 않은 고객의 고객이름을 검색해보자.

```
SELECT    고객이름
FROM      고객
WHERE     나이 IS NULL;
```

결과 테이블

	고객이름
1	오형준

데이터의 검색

NULL을 이용한 검색 (예제)

IS NULL / IS NOT NULL

예제 7-25

고객 테이블에서 나이가 이미 입력된 고객의 고객이름을 검색해보자.

```
SELECT    고객이름
FROM      고객
WHERE     나이 IS NOT NULL;
```

결과 테이블

	고객이름
1	정소화
2	김선우
3	고명석
4	김용욱
5	성원용
6	채광주

데이터의 검색

NULL 속성 조회 시 유의할 점

제품 테이블.재고량이 NULL 속성을 가지면 아래 연산 결과는 모두 False

```
재고량 > 10    →  False
```

```
재고량 = 10    →  False
```

```
재고량 <> 10   →  False
```

왜 모두 False일까?

NULL은 '값이 없음'이므로 어떤 비교 연산도 성립하지 않음 → 반드시 IS NULL / IS NOT NULL 사용

데이터의 검색

NULL 속성 조회 시 유의할 점 (계속)

제품 테이블.재고량이 NULL 속성을 가지면 아래 연산 결과는 모두 False

```
재고량 > 10    →  False
```

```
재고량 = 10    →  False
```

```
재고량 <> 10   →  False
```

결론

NULL 검색에는 반드시 IS NULL / IS NOT NULL을 사용해야 합니다.

실습 ④ IS NULL / IS NOT NULL

STEP 1 = NULL (틀린 방법!) — 따로 실행!

```
SELECT * FROM 고객 WHERE 나이 = NULL;
```

✗ 예상 결과: 에러!
0행! 아무것도 안 나옴! = NULL은 항상 False!

STEP 2 IS NULL (올바른 방법!) — 따로 실행!

```
SELECT * FROM 고객 WHERE 나이 IS NULL;
```

▶ 예상 결과
peach 1행! 이게 올바른 방법!

STEP 3 IS NOT NULL

```
SELECT * FROM 고객 WHERE 나이 IS NOT NULL;  -- 6명
```

데이터의 검색

정렬 검색

> **Default: ASC (오름차순)**

```
SELECT  [ ALL ¦ DISTINCT ] 속성_리스트
FROM    테이블_리스트
[ WHERE  조건 ]
[ ORDER BY  속성_리스트 [ ASC ¦ DESC ] ];
```


```
SELECT  [ ALL ¦ DISTINCT ] 속성_리스트
FROM    테이블_리스트
[ WHERE  조건 ]
[ ORDER BY  속성_리스트 [ ASC ¦ DESC ] ];
```

데이터의 검색

정렬 검색 (예제)

예제 7-26

고객 테이블에서 고객이름, 등급, 나이를 검색하되, 나이를 기준으로 내림차순 정렬해보자.

```
▶▶ SSELECT   고객이름, 등급, 나이
   FROM      고객
   ORDER BY  나이      DESC;
```

결과 테이블

	고객이름	등급	나이
1	오형준	silver	(null)
2	성원용	gold	35
3	채광주	silver	31
4	고명석	gold	28
5	김선우	vip	25
6	김용욱	silver	22
7	정소화	gold	20

데이터의 검색

정렬 검색 (예제)

예제 7-27

주문 테이블에서 수량이 10개 이상인 주문의 주문고객, 주문제품, 수량, 주문일자를 검색해 보자. 단, 주문제품을 기준으로 오름차순 정렬하고, 동일 제품은 수량을 기준으로 내림차순 정렬해보자.

```
▶▶ SELECT    주문고객, 주문제품, 수량, 주문일자
   FROM      주문
   WHERE     수량 >= 10
   ORDER BY  주문제품 ASC, 수량 DESC;
```

결과 테이블

	주문고객	주문제품	수량	주문일자
1	banana	p01	19	22/03/02
2	pear	p02	50	22/04/10
3	apple	p03	22	22/03/15
4	carrot	p03	20	22/05/22
5	apple	p03	10	22/01/01
6	banana	p04	15	22/04/11
7	banana	p06	45	22/01/11
8	melon	p06	36	22/02/20

실습 ⑤ ORDER BY 정렬

STEP 1 오름차순 (ASC = 기본값)

```
SELECT * FROM 고객 ORDER BY 나이;
```

STEP 2 내림차순 (DESC)

```
SELECT * FROM 주문 ORDER BY 수량 DESC;
```

STEP 3 복합 정렬 — 행 순서를 잘 보세요!

```
SELECT * FROM 고객 ORDER BY 등급 ASC, 적립금 DESC;
-- 등급 가나다순 → 같은 등급 안에서 적립금 높은 순
```

데이터의 검색

집계 함수(aggregate function)를 이용한 검색

> **열 함수(column function)라고도 함**

표 7-6 집계 함수

함수	의미	사용 가능한 속성의 타입
COUNT	속성 값의 개수	모든 데이터
MAX	속성 값의 최댓값	
MIN	속성 값의 최솟값	
SUM	속성 값의 합계	숫자 데이터
AVG	속성 값의 평균	

데이터의 검색

집계 함수 이용 시 유의할 사항

- 집계함수는 NULL 속성 값은 제외하고 계산함
- 집계함수는 WHERE 절에는 사용할 수 없음
- 집계함수는 SELECT 절이나 HAVING 절에서 사용할 수 있음

표 7-6 집계 함수

함수	의미	사용 가능한 속성의 타입
COUNT	속성 값의 개수	모든 데이터
MAX	속성 값의 최댓값	
MIN	속성 값의 최솟값	
SUM	속성 값의 합계	숫자 데이터
AVG	속성 값의 평균	

데이터의 검색

집계 함수 이용 시 유의할 사항

- 집계함수는 NULL 속성 값은 제외하고 계산함
- 집계함수는 WHERE 절에는 사용할 수 없음
- 집계함수는 SELECT 절이나 HAVING 절에서 사용할 수 있음

표 7-6 집계 함수

함수	의미	사용 가능한 속성의 타입
COUNT	속성 값의 개수	모든 데이터
MAX	속성 값의 최댓값	
MIN	속성 값의 최솟값	
SUM	속성 값의 합계	숫자 데이터
AVG	속성 값의 평균	

데이터의 검색

집계 함수 예제

예제 7-28

제품 테이블에서 모든 제품의 단가 평균을 검색해보자.

```
SELECT    AVG(단가)
FROM      제품;
```

결과 테이블

	AVG(단가)
1	2764.285714285714285714285714285714285714

데이터의 검색

집계 함수 예제

제품번호	제품명	재고량	단가	제조업체
p01	그냥만두	5000	4500	대한식품
p02	매운쫄면	2500	5500	민국푸드
p03	쿵떡파이	3600	2600	한빛제과
p04	맛난초콜릿	1250	2500	한빛제과
p05	얼큰라면	2200	1200	대한식품
p06	통통우동	1000	1550	민국푸드
p07	달콤비스킷	1650	1500	한빛제과

AVG(단가)

2764

그림 7-7 모든 제품의 평균 단가를 계산하는 과정 : 제품 테이블

데이터의 검색

집계 함수 예제 – 속성 별칭

> 쿼리 시 속성 별칭을 두는 것이 일반적임

예제 7-29

한빛제과에서 제조한 제품의 재고량 합계를 제품 테이블에서 검색해보자.

```
SELECT    SUM(재고량) AS "재고량 합계"
FROM      제품
WHERE     제조업체 = '한빛제과';
```

결과 테이블

	재고량 합계
1	6500

데이터의 검색

집계 함수 예제

예제 7-30

고객 테이블에 고객이 몇 명 등록되어 있는지 검색해보자.

▶▶ ❶ 고객아이디 속성을 이용해 계산하는 경우

```
SELECT   COUNT(고객아이디) AS 고객수
FROM     고객;
```

결과 테이블

	고객수
1	7

❷ 나이 속성을 이용해 계산하는 경우

```
SSELECT  COUNT(나이) AS 고객수
FROM     고객;
```

결과 테이블

	고객수
1	6

❸ *를 이용해 계산하는 경우

```
SELECT   COUNT(*) AS 고객수
FROM     고객;
```

결과 테이블

	고객수
1	7

데이터의 검색

집계 함수 – 고객 수 계산

고객 수 계산

고객아이디	고객이름	나이	등급	직업	적립금
apple	정소화	20	gold	학생	1000
banana	김선우	25	vip	간호사	2500
carrot	고명석	28	gold	교사	4500
orange	김용욱	22	silver	학생	0
melon	성원용	35	gold	회사원	5000
pear	채광주	31	silver	회사원	500
peach	오형준	NULL	silver	의사	300

COUNT (고객아이디) → 7

COUNT (나이) → 6

그림 7-8 고객의 수를 계산하는 과정 : 고객 테이블

데이터의 검색

집계 함수 — COUNT(*)

COUNT(*) 고객수 계산: 고객 테이블의 모든 튜플의 합

고객아이디	고객이름	나이	등급	직업	적립금
apple	정소화	20	gold	학생	1000
banana	김선우	25	vip	간호사	2500
carrot	고명석	28	gold	교사	4500
orange	김용욱	22	silver	학생	0
melon	성원용	35	gold	회사원	5000
pear	채광주	31	silver	회사원	500
peach	오형준	NULL	silver	의사	300

그림 7-9 COUNT(*)로 개수를 계산하는 과정 : 고객 테이블

데이터의 검색

집계 함수 — DISTINCT

DISTINCT를 이용한 집계

예제 7-31

제품 테이블에서 제조업체의 수를 검색해보자.

```
SELECT    COUNT(DISTINCT 제조업체) AS "제조업체 수"
FROM      제품;
```

결과 테이블

	제조업체 수
1	3

실습 ⑥ 집계 함수

STEP 1 COUNT(*) vs COUNT(속성) — 차이 비교!

```
SELECT COUNT(*) AS 전체고객수 FROM 고객;      -- 7
SELECT COUNT(나이) AS 나이있는수 FROM 고객;  -- 6 (NULL 제외!)
```

STEP 2 SUM, AVG, MAX, MIN 한꺼번에

```
SELECT SUM(단가) AS 합계, AVG(단가) AS 평균,
       MAX(단가) AS 최고, MIN(단가) AS 최저 FROM 제품;
```

STEP 3 WHERE + 집계함수

```
SELECT COUNT(*) AS 건수, SUM(수량) AS 총수량
  FROM 주문 WHERE 주문고객 = 'apple';
```

STEP 4 COUNT(DISTINCT ...)

```
SELECT COUNT(DISTINCT 제조업체) AS 업체수 FROM 제품;  -- 3
```

데이터의 검색

그룹별 검색

GROUP BY

HAVING

```
SELECT   [ ALL ¦ DISTINCT ] 속성_리스트
FROM     테이블_리스트
[ WHERE  조건 ]
[ GROUP BY  속성_리스트 [ HAVING 조건 ] ]
[ ORDER BY  속성_리스트 [ ASC ¦ DESC ] ];
```

예제 7-32

주문 테이블에서 주문제품별 수량의 합계를 검색해보자.

```
SELECT    주문제품, SUM(수량) AS 총주문수량
FROM      주문
GROUP BY  주문제품;
```

결과 테이블

	주문제품	총주문수량
1	p03	52
2	p02	58
3	p06	81
4	p04	15
5	p01	24

데이터의 검색

그룹별 검색

GROUP BY의 속성을 제외하고 집계 함수 사용 가능함

예제 7-32

주문 테이블에서 주문제품별 수량의 합계를 검색해보자.

```
▶▶ SELECT    주문제품, SUM(수량) AS 총주문수량
   FROM      주문
   GROUP BY  주문제품;
```

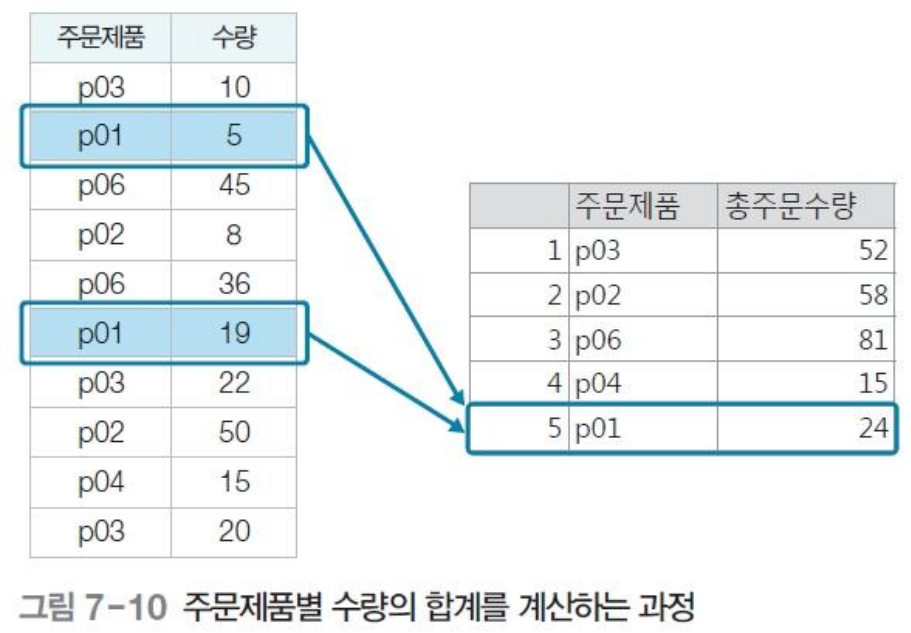
결과 테이블

	주문제품	총주문수량
1	p03	52
2	p02	58
3	p06	81
4	p04	15
5	p01	24

데이터의 검색

그룹별 검색 (결과)

GROUP BY의 속성을 제외하고 집계 함수 사용 가능함



주문제품	수량
p03	10
p01	5
p06	45
p02	8
p06	36
p01	19
p03	22
p02	50
p04	15
p03	20

	주문제품	총주문수량
1	p03	52
2	p02	58
3	p06	81
4	p04	15
5	p01	24

그림 7-10 주문제품별 수량의 합계를 계산하는 과정

데이터의 검색

그룹별 검색 (계속)

GROUP BY의 속성을 제외하고 쿼리도 가능함

예제 7-32

주문 테이블에서 주문제품별 수량의 합계를 검색해보자.

```
SELECT    주문제품, SUM(수량) AS 총주문수량
FROM      주문
GROUP BY  주문제품;
```

결과 테이블

	주문제품	총주문수량
1	p03	52
2	p02	58
3	p06	81
4	p04	15
5	p01	24

데이터의 검색

그룹별 검색 (계속)

GROUP BY의 속성을 제외하고 집계 함수 사용 가능함

예제 7-33

제품 테이블에서 제조업체별로 제조한 제품의 개수와 제품 중 가장 비싼 단가를 검색하되, 제품의 개수는 제품수라는 이름으로 출력하고 가장 비싼 단가는 최고가라는 이름으로 출력해 보자.

```
SELECT    제조업체, COUNT(*) AS 제품수, MAX(단가) AS 최고가
FROM      제품
GROUP BY  제조업체;
```

결과 테이블

	제조업체	제품수	최고가
1	대한식품	2	4500
2	민국푸드	2	5500
3	한빛제과	3	2600

데이터의 검색

그룹별 검색 — HAVING 조건

HAVING 조건 사용하기

예제 7-34

제품 테이블에서 제품을 3개 이상 제조한 제조업체별로 제품의 개수와, 제품 중 가장 비싼 단가를 검색해보자.

```
▶▶ SELECT    제조업체, COUNT(*) AS 제품수, MAX(단가) AS 최고가
   FROM      제품
   GROUP BY  제조업체  HAVING COUNT(*)>=3;
```

결과 테이블

	제조업체	제품수	최고가
1	한빛제과	3	2600

생략 가능

데이터의 검색

그룹별 검색 – HAVING 결과

예제 7-35

고객 테이블에서 적립금 평균이 1,000원 이상인 등급에 대해 등급별 고객수와 적립금 평균을 검색해보자.

```
SELECT    등급, COUNT(*) AS 고객수, AVG(적립금) AS 평균적립금
FROM      고객
GROUP BY  등급  HAVING AVG(적립금)>=1000;
```

결과 테이블

	등급	고객수	평균적립금
1	gold	3	3500
2	vip	1	2500

데이터의 검색

그룹별 검색 시 유의할 점

SELECT 절, GROUP BY 절 속성 개수를 동일하게 해야 함 (집계함수는 제외함)

예제 7-36

주문 테이블에서 각 주문고객이 주문한 제품의 총주문수량을 주문제품별로 검색해보자.

```
SELECT    주문제품, 주문고객, SUM(수량) AS 총주문수량
FROM      주문
GROUP BY  주문제품, 주문고객;
```

결과 테이블

	주문제품	주문고객	총주문수량
1	p02	carrot	8
2	p01	banana	19
3	p06	melon	36
4	p03	apple	32
5	p01	melon	5
6	p02	pear	50
7	p03	carrot	20
8	p06	banana	45
9	p04	banana	15

← 하나의 그룹

실습 ⑦ GROUP BY + HAVING

STEP 1 GROUP BY — 그룹별 집계

```
SELECT 주문제품, SUM(수량) AS 총수량
  FROM 주문 GROUP BY 주문제품;
```

STEP 2 HAVING — 그룹 조건 (STEP 1 결과와 비교!)

```
SELECT 주문제품, SUM(수량) AS 총수량
  FROM 주문 GROUP BY 주문제품
  HAVING SUM(수량) >= 30;
```

STEP 3 WHERE + GROUP BY + HAVING 종합

```
SELECT 주문제품, COUNT(*) AS 건수, SUM(수량) AS 총수량
  FROM 주문 WHERE 주문일자 >= '2022-02-01'
  GROUP BY 주문제품 HAVING COUNT(*) >= 2;
```

STEP 4 WHERE에 집계함수 쓰면? — 에러!

```
SELECT 주문제품, SUM(수량) FROM 주문
  WHERE SUM(수량) >= 30 GROUP BY 주문제품;
```

✗ 예상 결과: 에러!
WHERE에 집계함수 사용 불가! → HAVING을 써야!

Next Class

SQL을 이용한 데이터 조작 (계속)

여러 테이블에 대한 조인 검색, 부속 질의문,
INSERT, UPDATE, DELETE + 뷰, 삽입 SQL