

# Ch07. 데이터베이스 언어 SQL-3

SQL의 역할을 이해하고, 이를 기능별로 분류해본다.

SQL의 데이터 정의 기능을 예제를 통해 익힌다.

SQL의 데이터 조작 기능을 예제를 통해 익힌다.

뷰의 개념과 장점을 이해한다.

삽입 SQL의 역할을 이해한다



# 조인

## JOIN

주문번호에 따른 주문고객 이름과 배송지를 찾아라

SELECT 고객.고객아이디, 고객.고객이름, 주문.배송지  
FROM 고객  
JOIN 주문  
ON 고객.고객아이디 = 주문.주문고객

주문번호	고객이름	배송지
o01	정소화	서울시 마포구
o02	성원용	인천시 계양구
o03	김선우	경기도 부천시
o04	고명석	부산시 금정구
o05	성원용	경기도 용인시
o06	김선우	충청북도 보은군
o07	정소화	서울시 영등포구
o08	채광주	강원도 춘천시
o09	김선우	전라남도 목포시
o10	고명석	경기도 안양시

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	제고량	단가	제조업체
p01	그능만두	5000	4500	대한식품	
p02	매운쫄면	2500	5500	민국푸드	
p03	콩떡파이	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 7-2 질의에 사용할 판매 데이터베이스 : 고객, 제품, 주문 테이블

# 조인

## LEFT JOIN

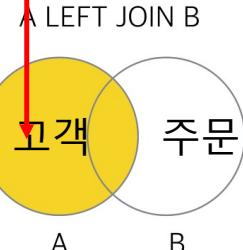
모든 고객들의 고객의 아이디, 이름 배송지를 검색하여라

SELECT 고객.고객아이디, 고객.고객이름, 주문.배송지  
FROM 고객

LEFT OUTER JOIN 주문

ON 고객.고객아이디 = 주문.주문고객

고객아이디	고객이름	배송지
apple	정소화	서울시 마포구
melon	성원용	인천시 계양구
banana	김선우	경기도 부천시
carrot	고명석	부산시 금정구
melon	성원용	경기도 용인시
banana	김선우	충청북도 보은군
apple	정소화	서울시 영등포구
pear	채광주	강원도 춘천시
banana	김선우	전라남도 목포시
carrot	고명석	경기도 안양시
orange	김용욱	NULL
peach	오형준	NULL



select  
from  
left outer join  
on

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

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 7-2 질의에 사용할 판매 데이터베이스 : 고객, 제품, 주문 테이블

# 조인

## RIGHT JOIN

모든 제품의 주문된 제품번호, 제품명, 주문고객을 검색하여라

SELECT 제품.제품번호, 제품.제품명, 주문.주문고객

FROM 주문

RIGHT OUTER JOIN 제품

ON 제품.제품번호 = 주문.주문제품

제품 번호	제품 명	주 문 고 객
p03	쿵 떡 파 이	apple
p01	그 낭 만 두	melon
p06	통 통 우 동	banana
p02	매 운 쫄 면	carrot
p06	통 통 우 동	melon
p01	그 낭 만 두	banana
p03	쿵 떡 파 이	apple
p02	매 운 쫄 면	pear
p04	맛 난 초 콜 릿	banana
p03	쿵 떡 파 이	carrot
p05	얼 큰 라 면	NULL
p07	달 콤 비 스 컷	NULL

A RIGHT JOIN B



Right outer join 지정

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 7-2 질의에 사용할 판매 데이터베이스 : 고객, 제품, 주문 테이블

# 조인

## LEFT JOIN

제품번호별 제품명, 주문고객의 수량, 주문고객의 나이, 등급을 검색하여라

SELECT 제품.제품번호, 제품.제품명, 주문.주문수량, 고객.나이, 고객.등급  
FROM 제품

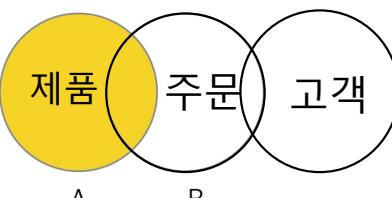
LEFT OUTER JOIN 주문

ON 제품.제품번호 = 주문.주문제품

LEFT OUTER JOIN 고객

ON 고객.고객아이디 = 주문.주문고객

A LEFT JOIN B



제품 번호	제품명	수량	나이	등급
p03	쿵 떡 파이	10	20	gold
p01	그 낭 만두	5	35	gold
p06	통 통 우동	45	25	vip
p02	매운쫄면	8	28	gold
p06	통 통 우동	36	35	gold
p01	그 낭 만두	19	25	vip
p03	쿵 떡 파이	22	20	gold
p02	매운쫄면	50	40	silver
p04	맛 난 초콜릿	15	25	vip
p03	쿵 떡 파이	20	28	gold
p05	얼큰라면	NULL	NULL	NULL
p07	달콤비스킷	NULL	NULL	NULL

12 rows in set (0.00 sec)

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	차광주	31	silver	회사원	500	

제품 테이블	제품번호	제품명	제고량	단가	제조업체
p01	그 낭 만두	5000	4500	대한식품	
p02	매운쫄면	2500	5500	민국푸드	
p03	쿵 떡파이	3600	2600	한빛제과	
p04	맛 난초콜릿	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 7-2 질의에 사용할 판매 데이터베이스 : 고객, 제품, 주문 테이블

# 조인

## LEFT JOIN

Apple 고객이 주문한 제품번호, 제품명, 수량, 나이, 등급을 검색하여라

SELECT 제품.제품번호, 제품.제품명, 주문.수량, 고객.나이, 고객.등급  
FROM 제품

LEFT OUTER JOIN 주문

ON 제품.제품번호 = 주문.주문제품

LEFT OUTER JOIN 고객

ON 고객.고객아이디 = 주문.주문고객

where 고객.고객아이디 ='apple'

제품번호	제품명	수량	나이	등급
p03	쿵떡파이	10	20	gold
p03	쿵떡파이	22	20	gold

2 rows in set (0.00 sec)

고객 테이블	고객아이디	고객이름	나이	등급	직업	직립금
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	그냥반두	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 영진구	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 7-2 질의에 사용할 판매 데이터베이스 : 고객, 제품, 주문 테이블

# MySQL 내장 함수

표 4-1 MySQL에서 제공하는 주요 내장 함수

구분	함수
단일행 함수	숫자 함수
	문자 함수(문자 반환)
	문자 함수(숫자 반환)
	날짜·시간 함수
	변환 함수
	정보 함수
	NULL 관련 함수
집계 함수	AVG, COUNT, MAX, MIN, STD, STDDEV, SUM
윈도우 함수(혹은 분석 함수)	CUME_DIST, DENSE_RANK, FIRST_VALUE, LAST_VALUE, LEAD, NTILE, RANK, ROW_NUMBER

# MySQL 내장 함수

함수	설명
<b>ABS(숫자)</b>	숫자의 절댓값을 계산 $\text{ABS}(-4.5) \Rightarrow 4.5$
<b>CEIL(숫자)</b>	숫자보다 크거나 같은 최소의 정수 $\text{CEIL}(4.1) \Rightarrow 5$
<b>FLOOR(숫자)</b>	숫자보다 작거나 같은 최소의 정수 $\text{FLOOR}(4.1) \Rightarrow 4$
<b>ROUND(숫자, m)</b>	숫자의 반올림, m은 반올림 기준 자릿수 $\text{ROUND}(5.36, 1) \Rightarrow 5.40$
<b>LOG(n, 숫자)</b>	숫자의 자연로그 값을 반환 $\text{LOG}(10) \Rightarrow 2.30259$
<b>POWER(숫자, n)</b>	숫자의 n제곱 값을 계산 $\text{POWER}(2, 3) \Rightarrow 8$
<b>SQRT(숫자)</b>	숫자의 제곱근 값을 계산(숫자는 양수) $\text{SQRT}(9.0) \Rightarrow 3.0$
<b>SIGN(숫자)</b>	수가 음수면 -1, 0이면 0, 양수면 1 $\text{SIGN}(3.45) \Rightarrow 1$

# MySQL 내장 함수

평균적립금

등급별 나이를 단위로 두번째 자리에서 반올림한 값을 구하시오.

```
SELECT 등급, ROUND(AVG(나이), 2) as avg_age  
FROM 고객  
GROUP BY 등급;
```

```
+-----+-----+  
| class | avg_age |  
+-----+-----+  
| gold  |  27.67 |  
| silver |  26.50 |  
| vip   |  25.00 |  
+-----+-----+  
3 rows in set (0.00 sec)
```

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

# MySQL 내장 함수

표 4-3 문자 함수의 종류

반환 구분	함수	설명
문자값 반환 함수	<b>CONCAT(s1,s2)</b>	두 문자열을 연결, CONCAT('마당', ' 서점') => '마당 서점'
	<b>LOWER(s)</b>	대상 문자열을 모두 소문자로 변환, LOWER('MR. SCOTT') => 'mr. scott'
	<b>LPAD(s,n,c)</b>	대상 문자열의 왼쪽부터 지정한 자리수까지 지정한 문자로 채움 LPAD('Page 1', 10, '*') => '*****Page 1'
	<b>REPLACE(s1,s2,s3)</b>	대상 문자열의 지정한 문자를 원하는 문자로 변경 REPLACE('JACK & JUE', 'J', 'BL') => 'BLACK & BLUE'
	<b>RPAD(s,n,c)</b>	대상 문자열의 오른쪽부터 지정한 자리수까지 지정한 문자로 채움 RPAD('AbC', 5, '*') => 'AbC**'
	<b>SUBSTR(s,n,k)</b>	대상 문자열의 지정된 자리에서부터 지정된 길이만큼 잘라서 반환 SUBSTR('ABCDEFG', 3, 4) => 'CDEF'
	<b>TRIM(c FROM s)</b>	대상 문자열의 양쪽에서 지정된 문자를 삭제(문자열만 넣으면 기본값으로 공백 제거) TRIM('==' FROM '==BROWNING==') => 'BROWNING'
	<b>UPPER(s)</b>	대상 문자열을 모두 대문자로 변환 UPPER('mr. scott') => 'MR. SCOTT'
숫자값 반환 함수	<b>ASCII(c)</b>	대상 알파벳 문자의 아스키 코드 값을 반환, ASCII('D') => 68
	<b>LENGTH(s)</b>	대상 문자열의 Byte 반환, 알파벳 1byte, 한글 3byte (UTF8) LENGTH('CANDIDE') => 7
	<b>CHAR_LENGTH(s)</b>	문자열의 문자 수를 반환, CHAR_LENGTH('데이터') => 3

# SQL 내장 함수

## REPLACE

고객테이블의 직업속성중 학생을 대학생으로 변경하시오

```
SELECT 고객아이디, REPLACE(직업, '학생', '대학생')
FROM 고객;
```

```
+-----+-----+
| 고객아이디 | 직업   |
+-----+-----+
| apple      | 대학생 |
| banana     | 간호사  |
| carrot     | 교사    |
| melon      | 회사원 |
| orange     | 대학생 |
| peach      | 의사    |
| pear       | 회사원 |
+-----+-----+
7 rows in set (0.00 sec)
```

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

# SQL 내장 함수

표 4-4 날짜·시간 함수의 종류

함수	반환형	설명
<b>STR_TO_DATE(string, format)</b>	DATE	문자열(STRING) 데이터를 날자형(DATE)으로 반환 STR_TO_DATE('2019-02-14', '%Y-%m-%d') => 2019-02-14
<b>DATE_FORMAT(date, format)</b>	STRING	날짜형(DATE) 데이터를 문자열(VARCHAR)로 반환 DATE_FORMAT('2019-02-14', '%Y-%m-%d') => '2019-02-14'
<b>ADDDATE(date, interval)</b>	DATE	DATE 형의 날짜에서 INTERVAL 지정한 시간만큼 더함 ADDDATE('2019-02-14', INTERVAL 10 DAY) => 2019-02-24
<b>DATE(date)</b>	DATE	DATE 형의 날짜 부분을 반환 SELECT DATE('2003-12-31 01:02:03'); => 2003-12-31
<b>DATEDIFF(date1, date2)</b>	INTEGER	DATE 형의 date1 – date2 날짜 차이를 반환 SELECT DATEDIFF('2019-02-14', '2019-02-04') => 10
<b>SYSDATE</b>	DATE	DBMS 시스템상의 오늘 날짜를 반환하는 함수 SYSDATE() => 2018-06-30 21:47:01

# SQL 내장 함수

주간 병우) 여기가 키

## ADDDATE

주문일로부터 10일 후 입금이 들어온다. 각 주문의 입금일자를 구하시오.

```
SELECT 주문번호, 주문고객, 주문일자,
       ADDDATE(주문일자, INTERVAL 10 DAY) as '입금일자'
  FROM 주문
```

주문번호	주문고객	주문일자	입금일자
o01	apple	2019-01-01	2019-01-11
o02	melon	2019-01-10	2019-01-20
o03	banana	2019-01-11	2019-01-21
o04	carrot	2019-02-01	2019-02-11
o05	melon	2019-02-20	2019-03-02
o06	banana	2019-03-02	2019-03-12
o07	apple	2019-03-15	2019-03-25
o08	pear	2019-04-10	2019-04-20
o09	banana	2019-04-11	2019-04-21
o10	carrot	2019-05-22	2019-06-01

10 rows in set (0.00 sec)

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 7-2 질의에 사용할 판매 데이터베이스 : 고객, 제품, 주문 테이블

# 04 뷰

## 뷰(View)

- 다른 테이블을 기반으로 만들어진 가상 테이블
  - 데이터를 실제로 저장하지 않고 논리적으로만 존재하는 테이블
- 일반 테이블과 동일한 방법으로 사용
- 뷰를 통해 기본 테이블의 내용을 쉽게 검색할 수는 있지만, 기본 테이블의 내용을 바꾸는 작업은 제한적으로 이루어짐
  - 기본 테이블: 뷰를 만드는데 기반이 되는 물리적인 테이블
- 다른 뷰를 기반으로 새로운 뷰를 만드는 것도 가능

## 04 뷰

### | 뷰는 기본 테이블을 들여 다 볼 수 있는 창의 역할을 담당

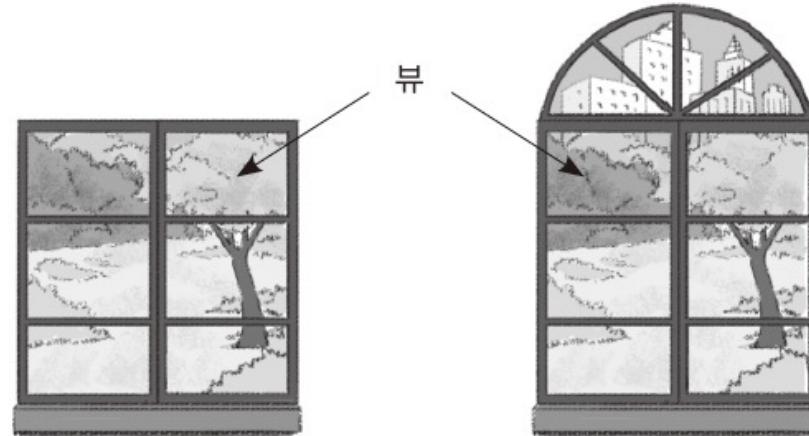


그림 7-12 뷰의 창 역할

# 04 뷰

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 7-13 뷰 예제에서 사용하는 판매 데이터베이스 : 고객, 제품, 주문 테이블

# 04 뷰

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

# 04 뷰

## 뷰 생성 : CREATE VIEW 문

```
CREATE VIEW 뷰_이름[(속성_리스트)]  
AS SELECT 문  
[WITH CHECK OPTION];
```

- CREATE VIEW 키워드와 함께 생성할 뷰의 이름과 속성의 이름을 나열
  - 속성 리스트를 생략하면 SELECT 절에 나열된 속성의 이름을 그대로 사용
- AS 키워드와 함께 기본 테이블에 대한 SELECT 문 제시
  - SELECT 문은 생성하려는 뷰의 정의를 표현하며 ORDER BY는 사용 불가
    - › 오라클과 같은 일부 DBMS에서는 ORDER BY를 허용하기도 함
- WITH CHECK OPTION
  - 뷰에 삽입이나 수정 연산을 할 때 SELECT 문에서 제시한 뷰의 정의 조건을 위반하면 수행되지 않도록 하는 제약조건을 지정

# 04 뷰

## 뷰 생성 : CREATE VIEW 문

예제 7-55

고객 테이블에서 등급이 vip인 고객의 고객아이디, 고객이름, 나이, 등급으로 구성된 뷰를 우수고객이라는 이름으로 생성해보자. 그런 다음 우수고객 뷰의 모든 내용을 검색해보자.

```
▶▶ CREATE VIEW 우수고객(고객아이디, 고객이름, 나이, 등급)
      AS SELECT 고객아이디, 고객이름, 나이, 등급
            FROM 고객
           WHERE 등급 = 'vip'
      WITH CHECK OPTION;

      SELECT * FROM 우수고객;
```

결과 테이블

	고객아이디	고객이름	나이	등급
1	banana	김선우	25	vip

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

뷰가 생성된 후에 우수고객 뷰에 vip 등급이 아닌 고객 데이터를 삽입하거나 뷰의 정의 조건을 위반하는 수정 및 삭제 연산을 시도하면 실행을 거부함  
(WITH CHECK OPTION 때문)

# 04 뷰

## 뷰 생성 : CREATE VIEW 문

```
CREATE VIEW    우수고객(고객아이디, 고객이름, 나이)
AS SELECT      고객아이디, 고객이름, 나이
               FROM    고객
               WHERE   등급 = 'vip'
               WITH CHECK OPTION;
```

```
=
```

```
CREATE VIEW    우수고객
AS SELECT      고객아이디, 고객이름, 나이
               FROM    고객
               WHERE   등급 = 'vip'
               WITH CHECK OPTION;
```

# 04 뷰

## 뷰 생성 : CREATE VIEW 문

### 예제 7-56

제품 테이블에서 제조업체별 제품수로 구성된 뷰를 업체별제품수라는 이름으로 생성해보자.  
그런 다음 업체별제품수 뷰의 모든 내용을 검색해보자.

```
▶▶ CREATE VIEW    업체별제품수(제조업체, 제품수)
      AS SELECT    제조업체, COUNT(*)
      FROM         제품
      GROUP BY    제조업체
      WITH CHECK OPTION;
```

```
SELECT * FROM 업체별제품수;
```

결과 테이블

	제조업체	제품수
1	대한식품	2
2	민국푸드	2
3	한빛제과	3

제품수 속성은 기본 테이블인  
제품 테이블에 원래 있던 속성이 아니라  
집계 함수를 통해 새로 계산된 것이므로  
속성의 이름을 명확히 제시해야 함

## 04 뷰

### 뷰 활용 : SELECT 문

- 뷰는 일반 테이블과 같은 방법으로 원하는 데이터를 검색할 수 있음
  - 뷰에 대한 SELECT 문이 내부적으로는 기본 테이블에 대한 SELECT 문으로 변환되어 수행
- 검색 연산은 모든 뷰에 수행 가능

#### 예제 7-57

우수고객 뷰에서 나이가 20세 이상인 고객에 대한 모든 내용을 검색해보자.

▶▶ `SELECT * FROM 우수고객 WHERE 나이 >= 20;`

결과 테이블

	고객아이디	고객이름	나이	등급
1	banana	김선우	25	vip

## 뷰 활용 : INSERT, UPDATE, DELETE 문

- 뷰에 대한 삽입·수정·삭제 연산 가능
  - 실제로 기본 테이블에 수행되므로 결과적으로는 기본 테이블이 변경됨
- 뷰에 대한 삽입·수정·삭제 연산은 제한적으로 수행됨
  - 변경 가능한 뷰 vs. 변경 불가능한 뷰

## 뷰 활용 : INSERT, UPDATE, DELETE 문

- 변경 불가능한 뷰의 특징
  - 기본 테이블의 기본키를 구성하는 속성이 포함되어 있지 않은 뷰
  - 기본 테이블에서 NOT NULL로 지정된 속성이 포함되어 있지 않은 뷰
  - 기본 테이블에 있던 내용이 아닌 집계 함수로 새로 계산된 내용을 포함하는 뷰
  - DISTINCT 키워드를 포함하여 정의한 뷰
  - GROUP BY 절을 포함하여 정의한 뷰
  - 여러 개의 테이블을 조인하여 정의한 뷰는 변경이 불가능한 경우가 많음

# 04 뷰

## 뷰 활용 : INSERT, UPDATE, DELETE 문

```
CREATE VIEW 제품1
AS SELECT 제품번호, 재고량, 제조업체
FROM 제품
WITH CHECK OPTION;

SELECT * FROM 제품1;
```

	제품번호	재고량	제조업체
1	p01	5000	대한식품
2	p02	2500	민국푸드
3	p03	3600	한빛제과
4	p04	1250	한빛제과
5	p05	2200	대한식품
6	p06	1000	민국푸드
7	p07	1650	한빛제과

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

제품1 뷰는 변경 가능한 뷰인가?

# 04 뷰

## 뷰 활용 : INSERT, UPDATE, DELETE 문

```
CREATE VIEW 제품2
AS SELECT 제품명, 재고량, 제조업체
FROM 제품
WITH CHECK OPTION;

SELECT * FROM 제품2;
```

	제품명	재고량	제조업체
1	그냥만두	5000	대한식품
2	매운쫄면	2500	민국푸드
3	쿵떡파이	3600	한빛제과
4	맛난초콜릿	1250	한빛제과
5	얼큰라면	2200	대한식품
6	통통우동	1000	민국푸드
7	달콤비스킷	1650	한빛제과

제품2 뷰는 변경 가능한 뷰인가?

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

# 04 뷰

## 뷰 활용 : INSERT, UPDATE, DELETE 문

### 예제 7-58

제품번호가 p08, 재고량이 1,000, 제조업체가 신선식품인 새로운 제품의 정보를 제품1 뷰에 삽입해보자. 그런 다음 제품1 뷰에 있는 모든 내용을 검색해보자.

▶▶ `INSERT INTO 제품1 VALUES ('p08', 1000, '신선식품');`

`SELECT * FROM 제품1;`

결과 테이블

	제품번호	재고량	제조업체
1	p01	5000	대한식품
2	p02	2500	민국푸드
3	p03	3600	한빛제과
4	p04	1250	한빛제과
5	p05	2200	대한식품
6	p06	1000	민국푸드
7	p07	1650	한빛제과
8	p08	1000	신선식품

# 04 뷰

## 뷰 활용 : INSERT, UPDATE, DELETE 문

SELECT \* FROM 제품;

	제품번호	제품명	재고량	단가	제조업체
1	p01	그냥만두	5000	4500	대한식품
2	p02	매운쫄면	2500	5500	민국푸드
3	p03	쿵떡파이	3600	2600	한빛제과
4	p04	맛난초콜릿	1250	2500	한빛제과
5	p05	얼큰라면	2200	1200	대한식품
6	p06	통통우동	1000	1550	민국푸드
7	p07	달콤비스킷	1650	1500	한빛제과
8	p08	(null)	1000	(null)	신선식품

제품1 뷰에 대한 삽입 연산은 실제로 기본 테이블인 제품 테이블에 수행된다.  
즉, 새로운 제품의 데이터는 제품 테이블에 삽입된다.

# 04 뷰

## 뷰 활용 : INSERT, UPDATE, DELETE 문

```
INSERT INTO 제품2 VALUES ('시원냉면', 1000, '신선식품');
```

제품2 뷰에 대한 삽입 연산은 실패함(오류 발생)

→ 제품2 뷰는 제품 테이블의 기본키인 제품번호 속성을 포함하고 있지 않기 때문에 제품2 뷰를 통해 새로운 투플을 삽입하려고 하면 제품번호 속성이 널 값이 되어 삽입 연산에 실패하게 됨

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

```
CREATE VIEW 제품2  
AS SELECT 제품명, 재고량, 제조업체  
FROM 제품  
WITH CHECK OPTION;
```

# 04 뷰

## 뷰의 장점

- 질의문을 좀 더 쉽게 작성할 수 있다.
  - GROUP BY, 집계 함수, 조인 등을 이용해 뷰를 미리 만들어 놓으면, 복잡한 SQL 문을 작성하지 않아도 SELECT 절과 FROM 절만으로도 원하는 데이터의 검색이 가능
- 데이터의 보안 유지에 도움이 된다.
  - 자신에게 제공된 뷰를 통해서만 데이터에 접근하도록 권한 설정이 가능
- 데이터를 좀 더 편리하게 관리할 수 있다.
  - 제공된 뷰와 관련이 없는 다른 내용에 대해 사용자가 신경 쓸 필요가 없음

## 뷰 삭제 : DROP VIEW 문

- 뷰를 삭제해도 기본 테이블은 영향을 받지 않음

```
DROP VIEW 뷰_이름;
```

- 만약, 삭제할 뷰를 참조하는 제약조건이 존재한다면?
  - 뷰 삭제가 수행되지 않음
  - 관련된 제약조건을 먼저 삭제해야 함
  - 예) 삭제할 뷰를 이용해 만들어진 다른 뷰가 존재하는 경우

## 04 뷰

### 뷰 삭제 : DROP VIEW 문

예제 7-59

우수고객 뷰를 삭제해보자.

▶▶ DROP VIEW 우수고객;

# 05 삽입 SQL

## ■ 삽입 SQL의 개념과 특징

- 삽입 SQL(ESQL; Embedded SQL)
  - 프로그래밍 언어로 작성된 응용 프로그램 안에 삽입하여 사용하는 SQL 문
- 주요 특징
  - 프로그램 안에서 일반 명령문이 위치할 수 있는 곳이면 어디든 삽입 가능
  - 일반 명령문과 구별하기 위해 삽입 SQL 문 앞에 EXEC SQL을 붙임
  - 프로그램에 선언된 일반 변수를 삽입 SQL 문에서 사용할 때는 이름 앞에 콜론(:)을 붙여서 구분함
- 커서(cursor)
  - 수행 결과로 반환된 여러 행을 한 번에 하나씩 가리키는 포인터
  - 여러 개의 행을 결과로 반환하는 SELECT 문을 프로그램에서 사용할 때 필요

# 05 삽입 SQL

## ■ 삽입 SQL 문에서 사용할 변수 선언 방법

- BEGIN DECLARE SECTION과 END DECLARE SECTION 사이에 선언

## ■ 커서가 필요 없는 삽입 SQL

- CREATE TABLE 문, INSERT 문, DELETE 문, UPDATE 문
- 결과로 행 하나만 반환하는 SELECT 문

04 B

## 고객 테이블

고객아이디	고객이름	나이	등급	직업	주급금
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



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

그림 7-13 뷰 예제에서 사용하는 편네 데이터베이스: 고객, 제품, 주문 테이블

- ① 주문수량이 30개 이상인 제품3 view 만들기  
 (Row 속성 (제품수량, 제품명, 단가))
- ② (100, 얼큰라면, 5000) insert

```

Order_Num      +-----+
| NULL | char(3) | NO | PR
Order_Customer | varchar(20) | YES | MU
Order_Product  | char(3) | YES | MU
amount         | int(11) | YES |
address        | varchar(30) | YES |
order_date     | date | YES |
+-----+
6 rows in set (0.00 sec)

mysql> desc Class_Product;
+-----+-----+-----+
| Field | Type   | Null | Key |
+-----+-----+-----+

```

```

CREATE VIEW PROD3
AS SELECT amount,Prod_Name,price
FROM Class_Order, Class_Product
WHERE Class_Order.Order_Product = Class_Product.Prod_ID
and amount >30;
Query OK, 0 rows affected (0.02 sec)

```

① 주문수량이 30개 이상인 제품3 view 만들기  
 (Row 속성 (제품수량, 제품명, 단가))  
 (100, 얼큰라면, 5000) insert

```

--> FROM Class_Order, Class_Product
--> WHERE Class_Order.Order_Product = Class_Product.Prod_ID
--> and amount >30;
Query OK, 0 rows affected (0.02 sec)

mysql> select * from PROD3;
+-----+-----+-----+
| amount | Prod_Name | price |
+-----+-----+-----+
| 45    | 통통우동  | 1550  |
| 36    | 통통우동  | 1550  |
| 50    | 매운쫄면  | 5500  |
+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>

```

view 만들기

↳ Drop view 이름

```

--> FROM Class_Order, Class_Product
--> WHERE Class_Order.Order_Product = Class_Product.Prod_ID
--> and amount >30;
Query OK, 0 rows affected (0.02 sec)

mysql> select * from PROD3;
+-----+-----+-----+
| amount | Prod_Name | price |
+-----+-----+-----+
| 45    | 통통우동  | 1550  |
| 36    | 통통우동  | 1550  |
| 50    | 매운쫄면  | 5500  |
+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>

```

# 05 삽입 SQL

삽입 SQL 문은 “EXEC SQL”을 붙여서 일반 명령문과 구분함

## C 언어

```
int main() {  
  
    EXEC SQL BEGIN DECLARE SECTION;  
    char p_no[4], p_name[21];  
    ❶ int price;  
    EXEC SQL END DECLARE SECTION;  
  
    printf("제품번호를 입력하세요 : ");  
    ❷ scanf("%s", p_no);  
  
    EXEC SQL SELECT 제품명, 단가 INTO :p_name, :price  
    ❸ FROM 제품  
    WHERE 제품번호 = :p_no;  
  
    ❹ printf("\n 제품명 = %s", p_name);  
    ❺ printf("\n 단가 = %d", price);  
  
    return 0;  
}
```

그림 7-14 입력된 제품번호에 해당되는 제품명과 단가를 검색하는 프로그램

# 05 삽입 SQL

## [그림 7-14]의 프로그램

- 입력된 제품번호에 해당하는 제품명과 단가를 검색하는 프로그램
- ① : 삽입 SQL 문에서 사용할 변수 선언
  - 테이블 내에 대응되는 속성과 같은 타입으로 변수의 데이터 타입을 선언
  - C 프로그램에서는 문자열의 끝을 표시하는 널 문자('₩0')을 포함할 수 있도록 변수 선언 시 대응되는 속성의 문자열 길이 보다 한 개 더 길게 선언
- ② : 검색하고자 하는 제품의 제품번호를 사용자로부터 입력받는 부분
- ③ : 제품 테이블에서 사용자가 입력한 제품번호에 해당하는 제품명과 단가를 검색하여 대응되는 각각의 변수에 저장하는 삽입 SQL 문
  - 변수는 INTO 키워드 다음에 차례대로 나열
- ④ : 검색된 제품명과 단가를 화면에 출력

# 05 삽입 SQL

## 커서가 필요한 삽입 SQL

- 커서를 선언하는 삽입 SQL 문
  - 커서를 사용하기 전에 커서의 이름과 커서가 필요한 SELECT 문 선언

```
EXEC SQL DECLARE 커서_이름 CURSOR FOR SELECT 문;
```

예) EXEC SQL DECLARE product\_cursor CURSOR FOR 제품 테이블에서 제품명과 단가를 모두 검색하는 SELECT 문을  
SELECT 제품명, 단가 FROM 제품; 위한 커서를 product\_cursor라는 이름으로 선언함

- 커서에 연결된 SELECT 문을 실행

```
EXEC SQL OPEN 커서_이름;
```

예) EXEC SQL OPEN product\_cursor; product\_cursor라는 이름의 커서에 연결된 SELECT 문을 실행함

# 05 삽입 SQL

## 커서가 필요한 삽입 SQL

- 커서를 이동시키는 삽입 SQL 문
  - 커서를 이동하여 처리할 다음 행을 가리키도록 하고, 커서가 가리키는 행에서 속성 값을 가져와 변수에 저장시킴
  - 결과 테이블에는 여러 행이 존재하므로 FETCH 문은 여러 번 수행해야 함
    - › for, while 문과 같은 반복문과 함께 사용

```
EXEC SQL FETCH 커서_이름 INTO 변수_리스트;
```

예) EXEC SQL FETCH product\_cursor INTO :p\_name, :price; product\_cursor 커서를 이동해 결과 테이블의 다음 행에 접근하여 제품명 속성의 값을 p\_name 변수에 저장하고 단가 속성의 값을 price 변수에 저장함

# 05 삽입 SQL

## 커서가 필요한 삽입 SQL

- 커서의 사용을 종료하는 삽입 SQL 문

```
EXEC SQL CLOSE 커서_이름;
```

예) EXEC SQL CLOSE product\_cursor; *product\_cursor 커서를 더는 사용하지 않음*

# PHP + SQL

## - 예제 2 소스

```
$conn = mysqli_connect("127.0.0.1", "root", "1234", "test_db");

$select_query = "SELECT seq, name FROM test_table WHERE seq <= 4";
$result_set = mysqli_query($conn, $select_query);

while ($row = mysqli_fetch_array($result_set)){
    print_r($row);
    echo '<br>';
}

mysqli_close($conn);
```

## - 결과

```
Array ( [0] => 1 [seq] => 1 [1] => 흥길동 [name] => 흥길동 )
Array ( [0] => 2 [seq] => 2 [1] => 일지매 [name] => 일지매 )
Array ( [0] => 3 [seq] => 3 [1] => 임꺽정 [name] => 임꺽정 )
Array ( [0] => 4 [seq] => 4 [1] => 이순신 [name] => 이순신 )
```

# Python

## Classnet 서버 내에서 접속하여 pandas 형태로 결과물 얻기

```
import pymysql
con = pymysql.connect(host='localhost', user='student', password='xxxxxx', db='employees', charset='utf8', autocommit=True, cursorclass=pymysql.cursors.DictCursor, port=33060)
cur = con.cursor()

sql = "select * from dept_manager" # customers 테이블 전체를 불러옴
cur.execute(sql)
rows = cur.fetchall()
con.close() # DB 연결 종료
print(rows)
```

이 부분은 본인  
아이디

DB

이 부분은 본인  
비밀번호

접속 후 → localhost 있는지 가하고 끝!

터미널 접속

# Python

## Classnet server

밖에서  
DB 드라이버

터미널 접속 X

```
from sshtunnel import SSHTunnelForwarder
import pymysql

server = SSHTunnelForwarder(
    ('classnet.mju.ac.kr', 1004),
    ssh_username='s학번',
    ssh_password='비밀번호',
    remote_bind_address=('127.0.0.1', 33060))

server.start()

mysqlcmd = pymysql.connect(user='s학번', port=server.local_bind_port, passwd='1234', host='127.0.0.1', charset='utf8')

sqlcur = mysqlcmd.cursor()
sqlcur.execute("use DBs학번")
sqlcur.execute("select * from XXX")

result = sqlcur.fetchall()
for data in result:
    print(data)
sqlcur.close()
```