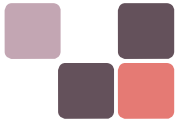
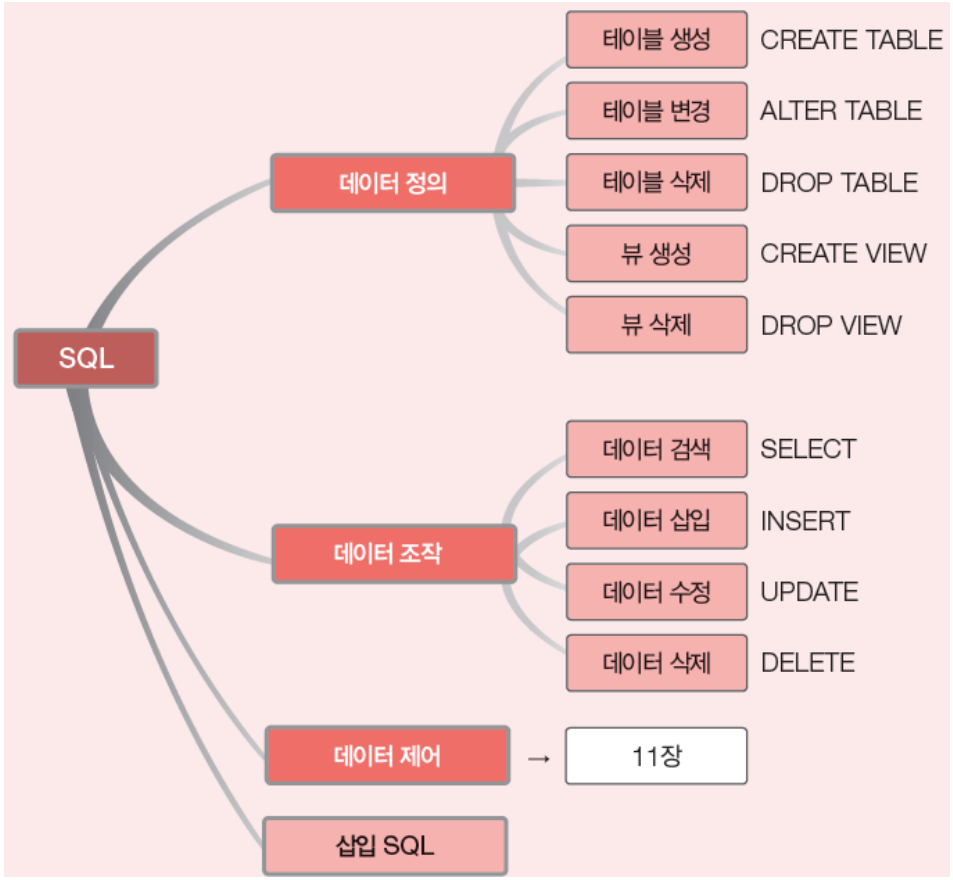


7장. 데이터베이스 언어 SQL

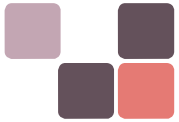
- SQL의 소개
- SQL을 이용한 데이터 정의
- SQL을 이용한 데이터 조작
- 뷰
- 삽입 SQL

학습목표



- 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.

01 SQL의 소개



❖ SQL(Structured Query Langauge)

- 의미
  - 관계 데이터베이스를 위한 표준 질의어
  - 비절차적 데이터 언어
- 발전 역사
  - SEQUEL(Structured English QUEry Language)에서 유래
    - SEQUEL : 연구용 관계 데이터베이스 관리 시스템인 SYSTEM R을 위한 언어
  - 미국 표준 연구소인 ANSI와 국제 표준화 기구인 ISO에서 표준화 작업을 진행
    - 계속 수정 및 보완되고 있음
- 사용 방식
  - 대화식 SQL : 데이터베이스 관리 시스템에 직접 접근해 질의를 작성하여 실행
  - 삽입 SQL : 프로그래밍 언어로 작성된 응용 프로그램에 삽입

01 SQL의 소개

❖ SQL의 분류

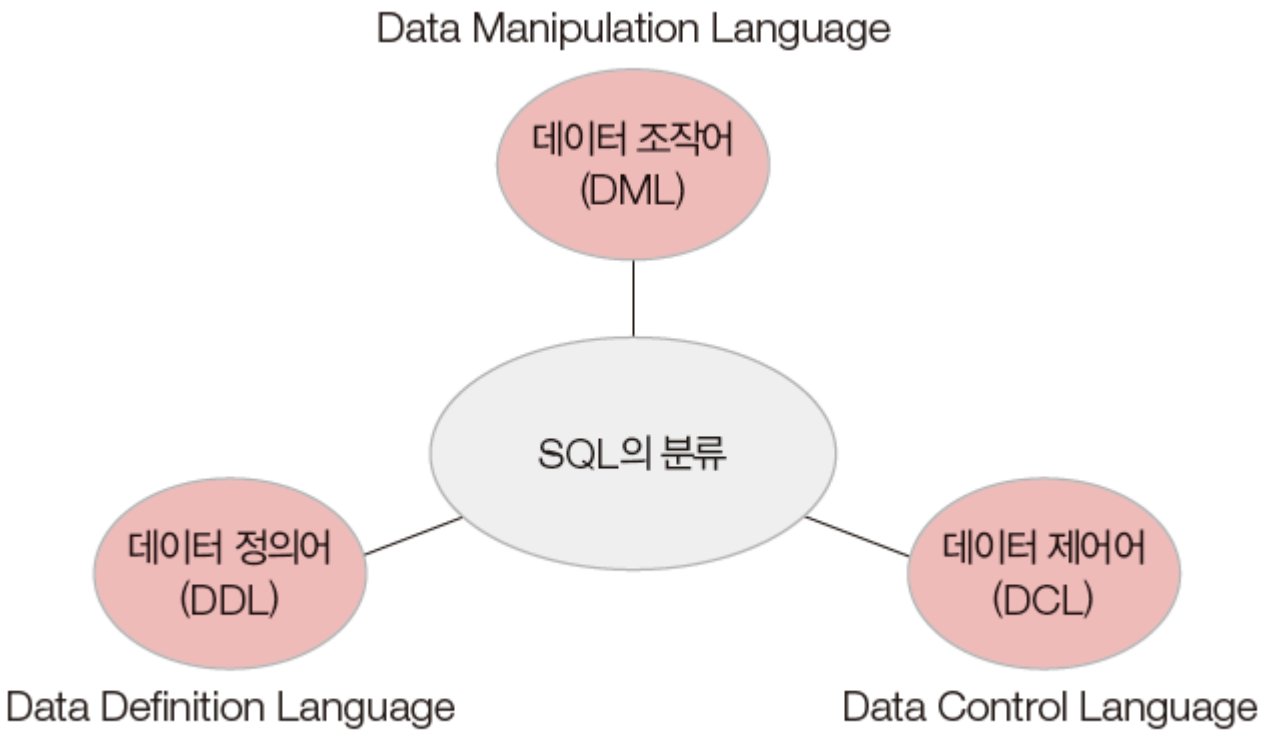
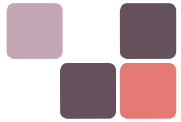


그림 7-1 SQL의 분류

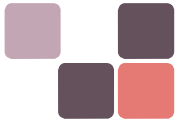
01 SQL의 소개



❖ SQL의 분류

- 데이터 정의어(DDL)
  - 테이블을 생성하고 변경·제거하는 기능을 제공
- 데이터 조작어(DML)
  - 테이블에 새 데이터를 삽입하거나, 테이블에 저장된 데이터를 수정·삭제·검색하는 기능을 제공
- 데이터 제어어(DCL)
  - 보안을 위해 데이터에 대한 접근 및 사용 권한을 사용자별로 부여하거나 취소하는 기능을 제공

01 SQL의 소개

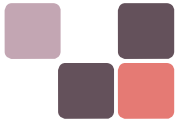


❖ 질의에 사용할 판매 데이터베이스 : 고객 테이블

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

01 SQL의 소개

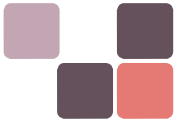


❖ 질의에 사용할 판매 데이터베이스 : 제품 테이블

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

01 SQL의 소개



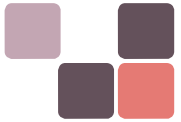
❖ 질의에 사용할 판매 데이터베이스 : 주문 테이블

주문 테이블

주문번호	주문고객	주문제품	수량	배송지	주문일자
o01	apple	p03	10	서울시 마포구	2019-01-01
o02	melon	p01	5	인천시 계양구	2019-01-10
o03	banana	p06	45	경기도 부천시	2019-01-11
o04	carrot	p02	8	부산시 금정구	2019-02-01
o05	melon	p06	36	경기도 용인시	2019-02-20
o06	banana	p01	19	충청북도 보은군	2019-03-02
o07	apple	p03	22	서울시 영등포구	2019-03-15
o08	pear	p02	50	강원도 춘천시	2019-04-10
o09	banana	p04	15	전라남도 목포시	2019-04-11
o10	carrot	p03	20	경기도 안양시	2019-05-22

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

02 SQL을 이용한 데이터 정의

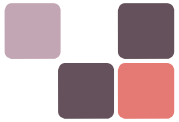


❖ SQL의 데이터 정의 기능

- 테이블 생성, 변경, 삭제

그림 7-3 SQL의 데이터 정의 기능

02 SQL을 이용한 데이터 정의

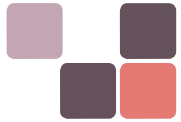


❖ 테이블 생성 : CREATE TABLE 문

```
CREATE TABLE 테이블_이름 (
        ❶ 속성_이름   데이터_타입  [NOT NULL]  [DEFAULT 기본_값]
        ❷ [PRIMARY KEY (속성_리스트)]
        ❸ [UNIQUE (속성_리스트)]
        ❹ [FOREIGN KEY (속성_리스트)  REFERENCES  테이블_이름(속성_리스트)]
           [ON DELETE 옵션]  [ON UPDATE 옵션]
        ❺ [CONSTRAINT 이름]  [CHECK(조건)]
);
```

- ❖ [ ]의 내용은 생략이 가능
- ❖ SQL 질의문은 세미콜론(;)으로 문장의 끝을 표시
- ❖ SQL 질의문은 대소문자를 구분하지 않음

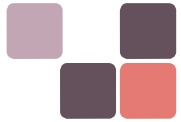
02 SQL을 이용한 데이터 정의



❖ 테이블 생성 : CREATE TABLE 문

- ① : 테이블을 구성하는 각 속성의 이름, 데이터 타입, 기본 제약 사항 정의
- ② : 기본키 정의
- ③ : 대체키 정의
- ④ : 외래키 정의
- ⑤ : 데이터 무결성을 위한 제약조건 정의

02 SQL을 이용한 데이터 정의

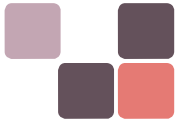


❖ 테이블 생성 : CREATE TABLE 문

- 속성의 정의
  - 테이블을 구성하는 각 속성의 데이터 타입을 선택한 다음 널 값 허용 여부와 기본 값 필요 여부를 결정
  - NOT NULL
    - 속성이 널 값을 허용하지 않음을 의미하는 키워드
    - 예) 고객아이디 VARCHAR(20) NOT NULL
  - DEFAULT
    - 속성의 기본 값을 지정하는 키워드
    - 예) 적립금 INT DEFAULT 0
    - 예) 담당자 VARCHAR(10) DEFAULT '방경아'

문자열이나 날짜 데이터는
작은 따옴표로 묶어서 표현
(작은 따옴표로 묶여진 문자열은
대소문자를 구분함)

02 SQL을 이용한 데이터 정의

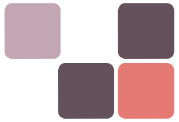


❖ 테이블 생성 : CREATE TABLE 문

표 7-1 속성의 데이터 타입

데이터 타입	의미
INT 또는 INTEGER	정수
SMALLINT	INT보다 작은 정수
CHAR(n) 또는 CHARACTER(n)	길이가 n인 고정 길이의 문자열
VARCHAR(n) 또는 CHARACTER VARYING(n)	최대 길이가 n인 가변 길이의 문자열
NUMERIC(p, s) 또는 DECIMAL(p, s)	고정 소수점 실수 p는 소수점을 제외한 전체 숫자의 길이고, s는 소수점 이하 숫자의 길이
FLOAT(n)	길이가 n인 부동 소수점 실수
REAL	부동 소수점 실수
DATE	연, 월, 일로 표현되는 날짜
TIME	시, 분, 초로 표현되는 시간
DATETIME	날짜와 시간

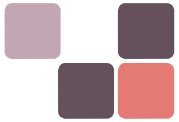
02 SQL을 이용한 데이터 정의



❖ 테이블 생성 : CREATE TABLE 문

- 키의 정의
  - PRIMARY KEY
    - 기본키를 지정하는 키워드
    - 예) PRIMARY KEY(고객아이디)
    - 예) PRIMARY KEY(주문고객, 주문제품)
  - UNIQUE
    - 대체키를 지정하는 키워드
    - 대체키로 지정되는 속성의 값은 유일성을 가지며 기본키와 달리 널 값이 허용됨
    - 예) UNIQUE(고객이름)

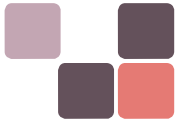
02 SQL을 이용한 데이터 정의



❖ 테이블 생성 : CREATE TABLE 문

- 키의 정의
  - FOREIGN KEY
    - 외래키를 지정하는 키워드
    - 외래키가 어떤 테이블의 무슨 속성을 참조하는지 REFERENCES 키워드 다음에 제시
    - 참조 무결성 제약조건 유지를 위해 참조되는 테이블에서 투플 삭제 시 처리 방법을 지정하는 옵션
      - » ON DELETE NO ACTION: 투플을 삭제하지 못하게 함
      - » ON DELETE CASCADE: 관련 투플을 함께 삭제함
      - » ON DELETE SET NULL: 관련 투플의 외래키 값을 NULL로 변경함
      - » ON DELETE SET DEFAULT: 관련 투플의 외래키 값을 미리 지정한 기본 값으로 변경함

02 SQL을 이용한 데이터 정의



❖ 테이블 생성 : CREATE TABLE 문

- 키의 정의
  - FOREIGN KEY
    - 참조 무결성 제약조건 유지를 위해 참조되는 테이블에서 투플 변경 시 처리 방법을 지정하는 옵션
      - » ON UPDATE NO ACTION: 투플을 변경하지 못하게 함
      - » ON UPDATE CASCADE : 관련 투플에서 외래키 값을 함께 변경함
      - » ON UPDATE SET NULL : 관련 투플의 외래키 값을 NULL로 변경함
      - » ON UPDATE SET DEFAULT : 관련 투플의 외래키 값을 미리 지정한 기본 값으로 변경함
    - 예) FOREIGN KEY(소속부서) REFERENCES 부서(부서번호)
    - 예) FOREIGN KEY(소속부서) REFERENCES 부서(부서번호)
      ON DELETE CASCADE ON UPDATE CASCADE

02 SQL을 이용한 데이터 정의

❖ 테이블 생성 : CREATE TABLE 문

[참조 무결성 제약조건 유지를 위한 투플 삭제 예]

부서 테이블

부서번호	부서이름
1	인사부
2	연구부
3	홍보부

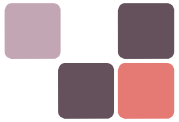
사원 테이블 (외래키)

사원번호	사원이름	소속부서
1001	정소화	3
1002	김용욱	1
1003	고명석	2

그림 7-4 외래키를 통해 관계를 맺고 있는 2개의 테이블

- ON DELETE NO ACTION: 부서 테이블의 투플을 삭제하지 못하게 함
- ON DELETE CASCADE : 사원 테이블에서 홍보부에 근무하는 정소화 사원 투플도 함께 삭제
- ON DELETE SET NULL : 사원 테이블에서 정소화 사원의 소속부서 속성 값을 NULL로 변경
- ON DELETE SET DEFAULT : 사원 테이블에서 정소화 사원의 소속부서 속성 값을 기본 값으로 변경

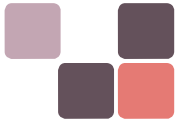
02 SQL을 이용한 데이터 정의



❖ 테이블 생성 : CREATE TABLE 문

- 데이터 무결성 제약조건의 정의
  - CHECK
    - 테이블에 정확하고 유효한 데이터를 유지하기 위해 특정 속성에 대한 제약조건을 지정
    - CONSTRAINT 키워드와 함께 고유의 이름을 부여할 수도 있음
    - 예) CHECK(재고량 >= 0 AND 재고량 <= 10000)
    - 예) CONSTRAINT CHK_CPY CHECK(제조업체 = '한빛제과')

02 SQL을 이용한 데이터 정의



❖ 고객 테이블 생성을 위한 CREATE TABLE 문 작성 예

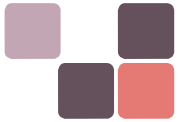
예제 7-1

고객 테이블은 고객아이디, 고객이름, 나이, 등급, 직업, 적립금 속성으로 구성되고, 고객아이디 속성이 기본키다. 고객이름과 등급 속성은 값을 반드시 입력해야 하고, 적립금 속성은 값을 입력하지 않으면 0이 기본으로 입력되도록 고객 테이블을 생성해보자.

```
▶▶ CREATE TABLE 고객 (
       고객아이디  VARCHAR(20)   NOT NULL,
       고객이름    VARCHAR(10)   NOT NULL,
       나이        INT,
       등급        VARCHAR(10)   NOT NULL,
       직업        VARCHAR(20),
       적립금      INT  DEFAULT 0,
       PRIMARY KEY(고객아이디)
   );
```

[고객 테이블(7 페이지)]

02 SQL을 이용한 데이터 정의



❖ 제품 테이블 생성을 위한 CREATE TABLE 문 작성 예

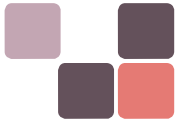
예제 7-2

제품 테이블은 제품번호, 제품명, 재고량, 단가, 제조업체 속성으로 구성되고, 제품번호 속성이 기본키다. 재고량이 항상 0개 이상 10,000개 이하를 유지하도록 제품 테이블을 생성해보자.

```
▶▶ CREATE TABLE 제품 (
       제품번호    CHAR(3)      NOT NULL,
       제품명      VARCHAR(20),
       재고량      INT,
       단가        INT,
       제조업체    VARCHAR(20),
       PRIMARY KEY(제품번호),
       CHECK (재고량 >= 0 AND 재고량 <=10000)
   );
```

[제품 테이블(8 페이지)]

02 SQL을 이용한 데이터 정의



❖ 주문 테이블 생성을 위한 CREATE TABLE 문 작성 예

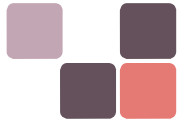
예제 7-3

주문 테이블은 주문번호, 주문고객, 주문제품, 수량, 배송지, 주문일자 속성으로 구성되고, 주문번호 속성이 기본키다. 주문고객 속성이 고객 테이블의 고객아이디 속성을 참조하는 외래키이고, 주문제품 속성이 제품 테이블의 제품번호 속성을 참조하는 외래키가 되도록 주문 테이블을 생성해보자.

```
▶▶ CREATE TABLE 주문 (
       주문번호    CHAR(3)      NOT NULL,
       주문고객    VARCHAR(20),
       주문제품    CHAR(3),
       수량        INT,
       배송지      VARCHAR(30),
       주문일자    DATE,
       PRIMARY KEY(주문번호),
       FOREIGN KEY(주문고객) REFERENCES 고객(고객아이디),
       FOREIGN KEY(주문제품) REFERENCES 제품(제품번호)
   );
```

[주문 테이블(9페이지)]

02 SQL을 이용한 데이터 정의



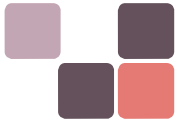
❖ 배송업체 테이블 생성을 위한 CREATE TABLE 문 작성 예

예제 7-4

배송업체 테이블은 업체번호, 업체명, 주소, 전화번호 속성으로 구성되고 업체번호 속성이 기본키다. 배송업체 테이블을 생성해보자.

```
▶▶ CREATE TABLE 배송업체 (
       업체번호    CHAR(3)       NOT NULL,
       업체명      VARCHAR(20),
       주소        VARCHAR(100),
       전화번호    VARCHAR(20),
       PRIMARY KEY(업체번호)
   );
```

02 SQL을 이용한 데이터 정의



❖ 테이블 변경 : ALTER TABLE 문

- 새로운 속성 추가

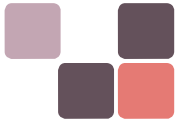
```
ALTER  TABLE  테이블_이름
    ADD  속성_이름  데이터_타입  [NOT NULL]  [DEFAULT 기본_값];
```

예제 7-5

[예제 7-1]에서 생성한 고객 테이블에 가입날짜 속성을 추가해보자.

```
▶▶ ALTER TABLE   고객   ADD   가입날짜   DATE;
```

02 SQL을 이용한 데이터 정의



❖ 테이블 변경 : ALTER TABLE 문

- 기존 속성 삭제

```
ALTER TABLE  테이블_이름  DROP COLUMN 속성_이름;
```

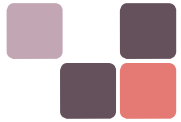
  - 만약, 삭제할 속성과 관련된 제약조건이 존재한다면?
    - 속성 삭제가 수행되지 않음
    - 관련된 제약조건을 먼저 삭제해야 함

예제 7-6

[예제 7-5]에서 추가한 고객 테이블의 가입날짜 속성을 삭제해보자.

```
▶▶ ALTER  TABLE  고객  DROP COLUMN  가입날짜;
```

02 SQL을 이용한 데이터 정의



❖ 테이블 변경 : ALTER TABLE 문

- 새로운 제약조건의 추가

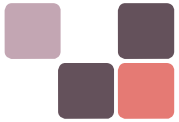
```
ALTER TABLE  테이블_이름  ADD  CONSTRAINT  제약조건_이름  제약조건_내용;
```

예제 7-7

고객 테이블에 20세 이상의 고객만 가입할 수 있다는 데이터 무결성 제약조건을 추가해보자.

```
▶▶ ALTER TABLE 고객 ADD CONSTRAINT CHK_AGE CHECK(나이 >= 20);
```

02 SQL을 이용한 데이터 정의



❖ 테이블 변경 : ALTER TABLE 문

- 기존 제약조건의 삭제

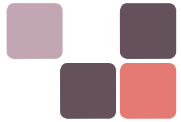
```
ALTER TABLE  테이블_이름  DROP  CONSTRAINT  제약조건_이름;
```

예제 7-8

[예제 7-7]에서 추가한 고객 테이블에 20세 이상의 고객만 가입할 수 있다는 데이터 무결성 제약조건을 삭제해보자.

```
▶▶ ALTER TABLE  고객  DROP CONSTRAINT  CHK_AGE;
```

02 SQL을 이용한 데이터 정의



❖ 테이블 삭제 : DROP TABLE 문

```
DROP TABLE  테이블_이름;
```

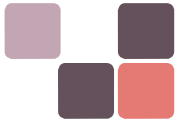
- 만약, 삭제할 테이블을 참조하는 테이블이 있다면?
  - 테이블 삭제가 수행되지 않음
  - 관련된 외래키 제약조건을 먼저 삭제해야 함

예제 7-9

[예제 7-4]에서 생성한 배송업체 테이블을 삭제해보자.

▶▶ DROP TABLE 배송업체;

03 SQL을 이용한 데이터 조작



❖ SQL의 데이터 조작 기능

- 데이터 검색, 새로운 데이터 삽입, 데이터 수정, 데이터 삭제

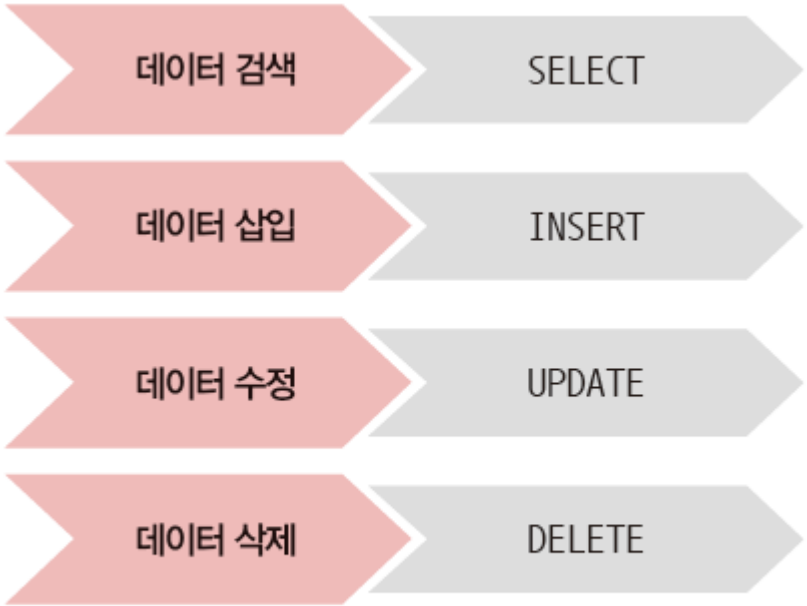
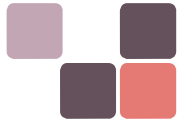


그림 7-5 SQL의 데이터 조작 기능

03 SQL을 이용한 데이터 조작

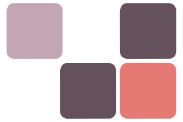


❖ 예제에서 사용할 판매 데이터베이스 : 고객 테이블

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

03 SQL을 이용한 데이터 조작

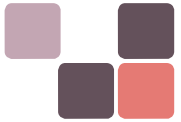


❖ 예제에서 사용할 판매 데이터베이스 : 제품 테이블

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

03 SQL을 이용한 데이터 조작



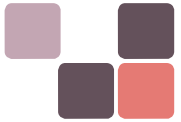
❖ 예제에서 사용할 판매 데이터베이스 : 주문 테이블

주문 테이블

주문번호	주문고객	주문제품	수량	배송지	주문일자
o01	apple	p03	10	서울시 마포구	2019-01-01
o02	melon	p01	5	인천시 계양구	2019-01-10
o03	banana	p06	45	경기도 부천시	2019-01-11
o04	carrot	p02	8	부산시 금정구	2019-02-01
o05	melon	p06	36	경기도 용인시	2019-02-20
o06	banana	p01	19	충청북도 보은군	2019-03-02
o07	apple	p03	22	서울시 영등포구	2019-03-15
o08	pear	p02	50	강원도 춘천시	2019-04-10
o09	banana	p04	15	전라남도 목포시	2019-04-11
o10	carrot	p03	20	경기도 안양시	2019-05-22

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

03 SQL을 이용한 데이터 조작



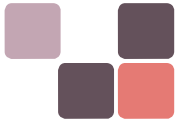
❖ 데이터 검색 : SELECT 문

- 기본 검색
  - SELECT 키워드와 함께 검색하고 싶은 속성의 이름을 나열
  - FROM 키워드와 함께 검색하고 싶은 속성이 있는 테이블의 이름을 나열
  - 검색 결과는 테이블 형태로 반환됨

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

  - ALL
    - 결과 테이블이 투플의 중복을 허용하도록 지정, 생략 가능
  - DISTINCT
    - 결과 테이블이 투플의 중복을 허용하지 않도록 지정

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 기본 검색

예제 7-10

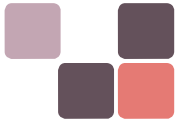
고객 테이블에서 고객아이디, 고객이름, 등급 속성을 검색해보자.

```
▶▶ 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

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 기본 검색

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

03 SQL을 이용한 데이터 조작

❖ 데이터 검색 : SELECT 문

- 기본 검색

모든 속성을 검색할 때는
모든 속성의 이름을 나열하지 않고
* 사용 가능

예제 7-12

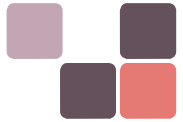
고객 테이블에 존재하는 모든 속성을 검색해보자.

```
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

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 기본 검색

예제 7-13

제품 테이블에서 제조업체를 검색해보자.

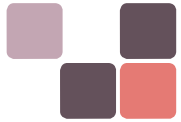
```
SELECT    제조업체
FROM      제품;
```

결과 테이블

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

결과 테이블에서 제조업체가 중복됨

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 기본 검색

예제 7-14

제품 테이블에서 제조업체를 검색하되, ALL 키워드를 사용해보자.

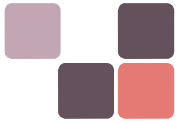
```
▶▶ SELECT    ALL    제조업체
   FROM      제품;
```

결과 테이블

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

결과 테이블에서 제조업체가 중복됨

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

▪ 기본 검색

예제 7-15

제품 테이블에서 제조업체 속성을 중복 없이 검색해보자.

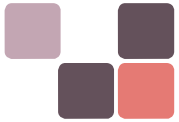
```
SELECT    DISTINCT    제조업체
FROM      제품;
```

결과 테이블

	제조업체
1	대한식품
2	민국푸드
3	한빛제과

결과 테이블에서 제조업체가 한 번씩만 나타남

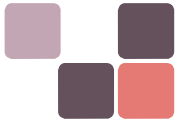
03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 기본 검색
  - AS 키워드를 이용해 결과 테이블에서 속성의 이름을 바꾸어 출력 가능
    - 새로운 이름에 공백이 포함되어 있으면 큰따옴표나 작은따옴표로 묶어주어야 함
      - » 오라클에서는 큰따옴표, MS SQL 서버에서는 작은따옴표 사용
    - AS 키워드는 생략 가능

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

▪ 기본 검색

예제 7-16

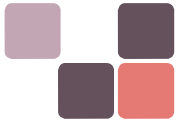
제품 테이블에서 제품명과 단가를 검색하되, 단가를 가격이라는 새 이름으로 출력해보자.

```
▶▶ 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

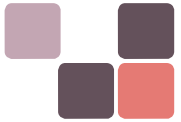
03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 산술식을 이용한 검색
  - SELECT 키워드와 함께 산술식 제시
    - 산술식: 속성의 이름과 +, -, *, / 등의 산술 연산자와 상수로 구성
  - 속성의 값이 실제로 변경되는 것은 아니고 결과 테이블에서만 계산된 결과 값이 출력됨

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 산술식을 이용한 검색

예제 7-17

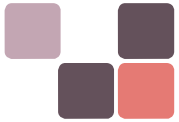
제품 테이블에서 제품명과 단가 속성을 검색하되, 단가에 500원을 더해 '조정 단가'라는 새 이름으로 출력해보자.

```
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

03 SQL을 이용한 데이터 조작



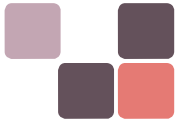
❖ 데이터 검색 : SELECT 문

- 조건 검색
  - 조건을 만족하는 데이터만 검색

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

  - WHERE 키워드와 함께 비교 연산자와 논리 연산자를 이용한 검색 조건 제시
    - 숫자뿐만 아니라 문자나 날짜 값을 비교하는 것도 가능
      - » 예) 'A' < 'C'
      - » 예) '2019-12-01' < '2019-12-02'
    - 조건에서 문자나 날짜 값은 작은따옴표로 묶어서 표현

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 조건 검색

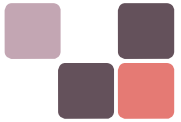
표 7-2 비교 연산자

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

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.

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 조건 검색

예제 7-18

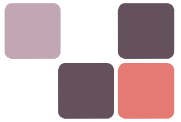
제품 테이블에서 한빛제과가 제조한 제품의 제품명, 재고량, 단가를 검색해보자.

```
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

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 조건 검색

예제 7-19

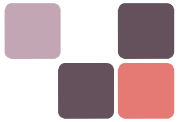
주문 테이블에서 apple 고객이 15개 이상 주문한 주문제품, 수량, 주문일자를 검색해보자.

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

결과 테이블

	주문제품	수량	주문일자
1	p03	22	19/03/15

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

▪ 조건 검색

예제 7-20

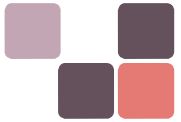
주문 테이블에서 apple 고객이 주문했거나 15개 이상 주문된 제품의 주문제품, 수량, 주문일자, 주문고객을 검색해보자.

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

결과 테이블

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

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

▪ 조건 검색

예제 7-21

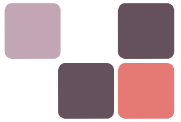
제품 테이블에서 단가가 2,000원 이상이면서 3,000원 이하인 제품의 제품명, 단가, 제조업체를 검색해보자.

```
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

03 SQL을 이용한 데이터 조작

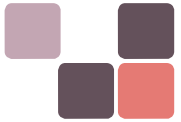


❖ 데이터 검색 : SELECT 문

- LIKE를 이용한 검색
  - LIKE 키워드를 이용해 부분적으로 일치하는 데이터를 검색
  - 문자열을 이용하는 조건에만 LIKE 키워드 사용 가능

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)



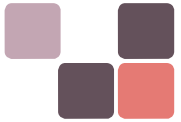
❖ 데이터 검색 : SELECT 문

- LIKE를 이용한 검색

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

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- LIKE를 이용한 검색

예제 7-22

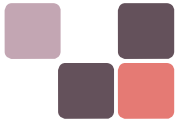
고객 테이블에서 성이 김 씨인 고객의 고객이름, 나이, 등급, 적립금을 검색해보자.

```
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

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- LIKE를 이용한 검색

예제 7-23

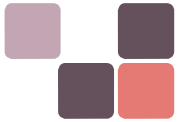
고객 테이블에서 고객아이디가 5자인 고객의 고객아이디, 고객이름, 등급을 검색해보자.

```
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

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- NULL을 이용한 검색
  - IS NULL 키워드를 이용해 특정 속성의 값이 널 값인지를 비교
  - IS NOT NULL 키워드를 이용해 특정 속성의 값이 널 값이 아닌지를 비교
  - 검색 조건에서 널 값은 다른 값과 크기를 비교하면 결과가 모두 거짓이 됨

예제 7-24

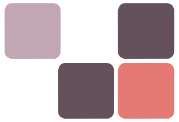
고객 테이블에서 나이가 아직 입력되지 않은 고객의 고객이름을 검색해보자.

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

결과 테이블

	고객이름
1	오형준

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- NULL을 이용한 검색

예제 7-25

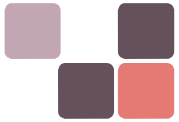
고객 테이블에서 나이가 이미 입력된 고객의 고객이름을 검색해보자.

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

결과 테이블

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

03 SQL을 이용한 데이터 조작

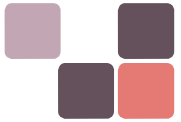


❖ 데이터 검색 : SELECT 문

- 정렬 검색
  - ORDER BY 키워드를 이용해 결과 테이블 내용을 사용자가 원하는 순서로 출력
  - ORDER BY 키워드와 함께 정렬 기준이 되는 속성과 정렬 방식을 지정
    - 오름차순(기본): ASC / 내림차순: DESC
    - 널 값은 오름차순에서는 맨 마지막에 출력되고, 내림차순에서는 맨 먼저 출력됨
    - 여러 기준에 따라 정렬하려면 정렬 기준이 되는 속성을 차례대로 제시

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

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 정렬 검색

예제 7-26

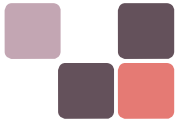
고객 테이블에서 고객이름, 등급, 나이를 검색하되, 나이를 기준으로 내림차순 정렬해보자.

```
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

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

▪ 정렬 검색

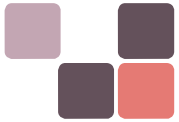
예제 7-27

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

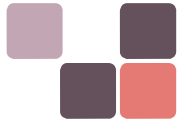
	주문고객	주문제품	수량	주문일자
1	banana	p01	19	19/03/02
2	pear	p02	50	19/04/10
3	apple	p03	22	19/03/15
4	carrot	p03	20	19/05/22
5	apple	p03	10	19/01/01
6	banana	p04	15	19/04/11
7	banana	p06	45	19/01/11
8	melon	p06	36	19/02/20



❖ 데이터 검색 : SELECT 문

- 집계 함수를 이용한 검색
  - 특정 속성 값을 통계적으로 계산한 결과를 검색하기 위해 집계 함수를 이용
    - 집계 함수(aggregate function)
      - » 열 함수(column function)라고도 함
      - » 개수, 합계, 평균, 최댓값, 최솟값의 계산 기능을 제공
  - 집계 함수 사용 시 주의 사항
    - 집계 함수는 널인 속성 값은 제외하고 계산함
    - 집계 함수는 WHERE 절에서는 사용할 수 없고, SELECT 절이나 HAVING 절에서만 사용 가능

03 SQL을 이용한 데이터 조작



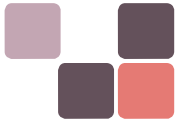
❖ 데이터 검색 : SELECT 문

- 집계 함수를 이용한 검색

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	

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 집계 함수를 이용한 검색

예제 7-28

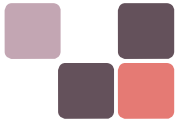
제품 테이블에서 모든 제품의 단가 평균을 검색해보자.

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

결과 테이블

	AVG(단가)
1	2764.285714285714285714285714285714285714

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 집계 함수를 이용한 검색

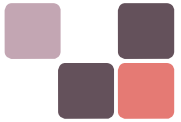
제품번호	제품명	재고량	단가	제조업체
p01	그냥만두	5000	4500	대한식품
p02	매운쫄면	2500	5500	민국푸드
p03	쿵떡파이	3600	2600	한빛제과
p04	맛난초콜릿	1250	2500	한빛제과
p05	얼큰라면	2200	1200	대한식품
p06	통통우동	1000	1550	민국푸드
p07	달콤비스킷	1650	1500	한빛제과

AVG(단가)

2764

그림 7-7 모든 제품의 평균 단가를 계산하는 과정 : 제품 테이블

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 집계 함수를 이용한 검색

예제 7-29

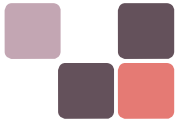
한빛제과에서 제조한 제품의 재고량 합계를 제품 테이블에서 검색해보자.

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

결과 테이블

	재고량 합계
1	6500

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 집계 함수를 이용한 검색

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

03 SQL을 이용한 데이터 조작

데이터 검색 : SELECT 문

- 집계 함수를 이용한 검색

널인 속성 값은 제외하고 개수 계산

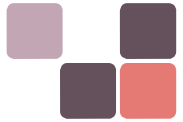
고객아이디	고객이름	나이	등급	직업	적립금
apple	정소화	20	gold	학생	1000
banana	김선우	25	vip	간호사	2500
carrot	고명석	28	gold	교사	4500
orange	김용욱	22	silver	학생	0
melon	성원용	35	gold	회사원	5000
pear	채광주	31	silver	회사원	500
peach	오형준	NULL	silver	의사	300

COUNT (고객아이디) → 7

COUNT (나이) → 6

그림 7-8 고객의 수를 계산하는 과정 : 고객 테이블

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

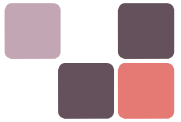
- 집계 함수를 이용한 검색

정확한 개수를 계산하기 위해서는
보통 기본키 속성이나 *를 주로 이용

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

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 집계 함수를 이용한 검색

예제 7-31

제품 테이블에서 제조업체의 수를 검색해보자.

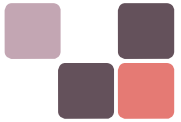
```
SELECT    COUNT(DISTINCT 제조업체) AS "제조업체 수"
FROM      제품;
```

결과 테이블

	제조업체 수
1	3

DISTINCT 키워드를 이용해 중복을 없애고 서로 다른 제조업체의 개수만 계산

03 SQL을 이용한 데이터 조작



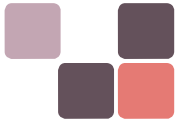
❖ 데이터 검색 : SELECT 문

- 그룹별 검색

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

- GROUP BY 키워드를 이용해 특정 속성의 값이 같은 투플을 모아 그룹을 만들고, 그룹별로 검색
  - GROUP BY 키워드와 함께 그룹을 나누는 기준이 되는 속성을 지정
- HAVING 키워드를 함께 이용해 그룹에 대한 조건을 작성
- 그룹을 나누는 기준이 되는 속성을 SELECT 절에도 작성하는 것이 좋음

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 그룹별 검색

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

데이터 검색 : SELECT 문

- 그룹별 검색

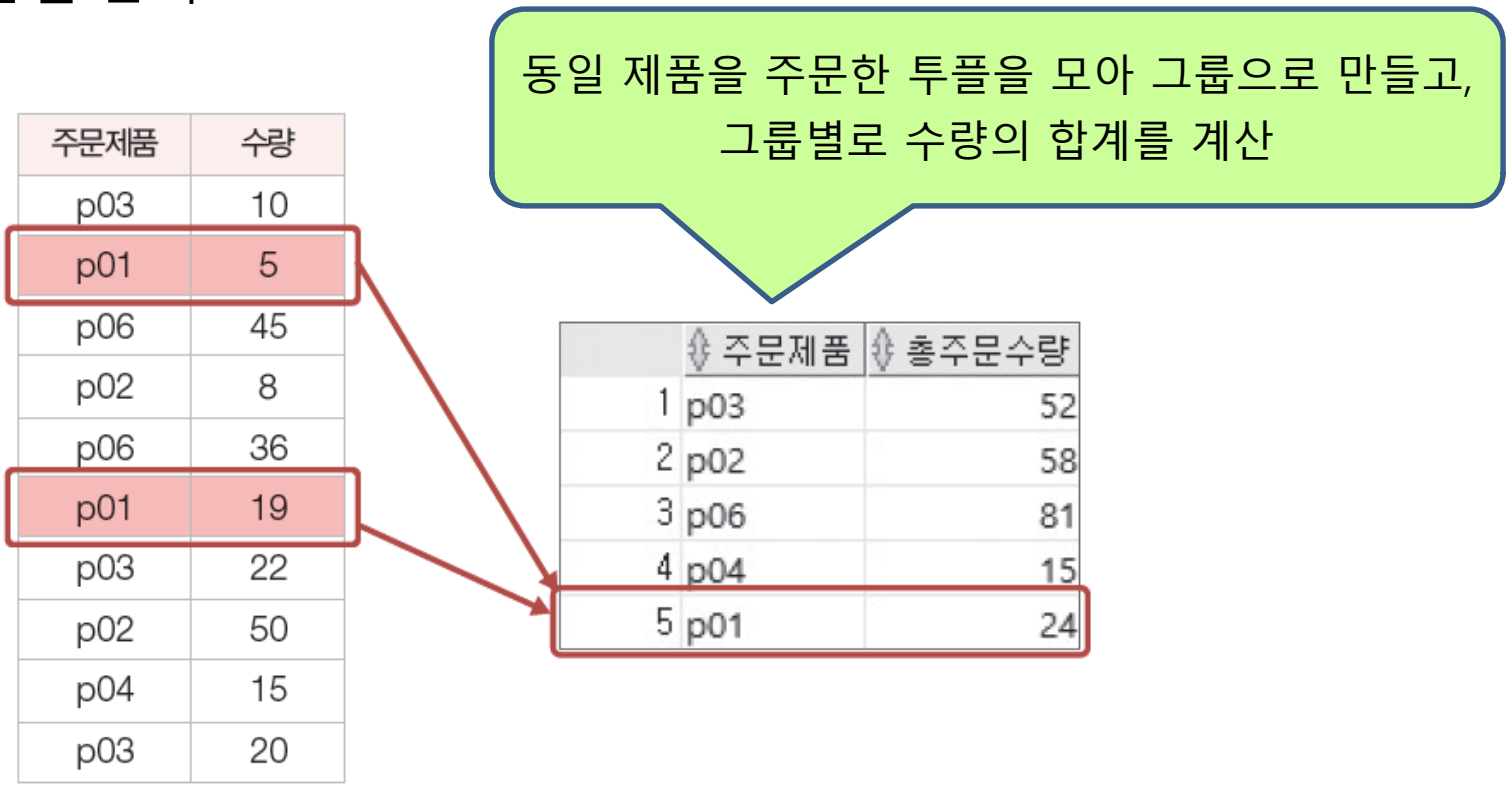
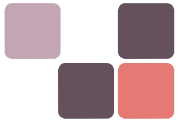


그림 7-10 주문제품별 수량의 합계를 계산하는 과정

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 그룹별 검색

예제 7-33

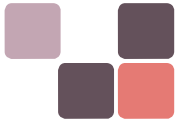
제품 테이블에서 제조업체별로 제조한 제품의 개수와 제품 중 가장 비싼 단가를 검색하되, 제품의 개수는 제품수라는 이름으로 출력하고 가장 비싼 단가는 최고가라는 이름으로 출력해 보자.

```
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

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

예제 7-34

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

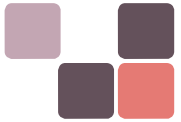
```
SELECT    제조업체, COUNT(*) AS 제품수, MAX(단가) AS 최고가
FROM      제품
GROUP BY  제조업체  HAVING COUNT(*)>=3;
```

결과 테이블

	제조업체	제품수	최고가
1	한빛제과	3	2600

집계 함수를 이용한 조건은 WHERE 절에는 작성할 수 없고 HAVING 절에서 작성

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 그룹별 검색

예제 7-35

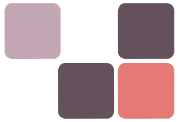
고객 테이블에서 적립금 평균이 1,000원 이상인 등급에 대해 등급별 고객수와 적립금 평균을 검색해보자.

```
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

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

▪ 그룹별 검색

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;
```

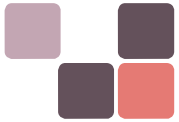
결과 테이블

	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

집계 함수나 GROUP BY 절에 명시된 속성 외의 속성은 SELECT 절에 작성 불가

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 여러 테이블에 대한 조인 검색
  - 조인 검색: 여러 개의 테이블을 연결하여 데이터를 검색하는 것
  - 조인 속성: 조인 검색을 위해 테이블을 연결해주는 속성
    - 연결하려는 테이블 간에 조인 속성의 이름은 달라도 되지만 도메인은 같아야 함
    - 일반적으로 외래키를 조인 속성으로 이용함
  - FROM 절에 검색에 필요한 모든 테이블을 나열
  - WHERE 절에 조인 속성의 값이 같아야 함을 의미하는 조인 조건을 제시
  - 같은 이름의 속성이 서로 다른 테이블에 존재할 수 있기 때문에 속성 이름 앞에 해당 속성이 소속된 테이블의 이름을 표시

    예) 주문.주문고객

03 SQL을 이용한 데이터 조작

데이터 검색 : SELECT 문

- 여러 테이블에 대한 조인 검색

외래키

주문 테이블

주문번호	주문고객	주문제품	수량	배송지	주문일자
o01	apple	p03	10	서울시 마포구	2019-01-01
o02	melon	p01	5	인천시 계양구	2019-01-10
o03	banana	p06	45	경기도 부천시	2019-01-11
o04	carrot	p02	8	부산시 금정구	2019-02-01
o05	melon	p06	36	경기도 용인시	2019-02-20
o06	banana	p01	19	충청북도 보은군	2019-03-02
o07	apple	p03	22	서울시 영등포구	2019-03-15
o08	pear	p02	50	강원도 춘천시	2019-04-10
o09	banana	p04	15	전라남도 목포시	2019-04-11
o10	carrot	p03	20	경기도 안양시	2019-05-22

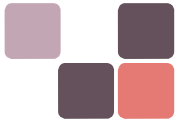
제품 테이블

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

조인 속성

그림 7-11 2개의 테이블을 이용한 조인 검색 예 : 주문과 제품 테이블

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 여러 테이블에 대한 조인 검색

예제 7-37

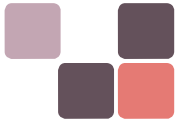
판매 데이터베이스에서 banana 고객이 주문한 제품의 이름을 검색해보자.

```
SELECT    제품.제품명
FROM      제품, 주문
WHERE     주문.주문고객 = 'banana'  AND  제품.제품번호 = 주문.주문제품;
```

결과 테이블

	제품명
1	그냥만두
2	맛난초콜릿
3	통통우동

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 여러 테이블에 대한 조인 검색

예제 7-38

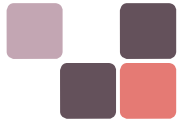
판매 데이터베이스에서 나이가 30세 이상인 고객이 주문한 제품의 주문제품과 주문일자를 검색해보자.

```
SELECT    주문.주문제품, 주문.주문일자
FROM      고객, 주문
WHERE     고객.나이 >= 30  AND  고객.고객아이디 = 주문.주문고객;
```

결과 테이블

	주문제품	주문일자
1	p01	19/01/10
2	p06	19/02/20
3	p02	19/04/10

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

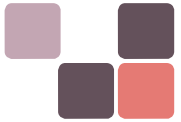
- 여러 테이블에 대한 조인 검색

이름이 같은 속성이 없다면
테이블 이름 없이 속성 이름으로만 작성해도 된다.

```
SELECT   주문제품, 주문일자
FROM     고객 c, 주문 o
WHERE    c.나이 >= 30 AND  o.주문고객 = c.고객아이디;
```

테이블의 이름을 대신하는 단순한 별명을 제시하여
질의문을 작성하는 것도 좋다.

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 여러 테이블에 대한 조인 검색

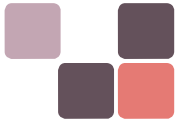
예제 7-39

판매 데이터베이스에서 고명석 고객이 주문한 제품의 제품명을 검색해보자.

```
SELECT    제품.제품명
FROM      고객, 제품, 주문
WHERE     고객.고객이름 = '고명석'  AND  고객.고객아이디 = 주문.주문고객
              AND  제품.제품번호 = 주문.주문제품;
```

결과 테이블

	제품명
1	매운쫄면
2	쿵떡파이



❖ 데이터 검색 : SELECT 문

- 부속 질의문을 이용한 검색
  - SELECT 문 안에 또 다른 SELECT 문을 포함하는 질의
    - 상위 질의문(주 질의문): 다른 SELECT 문을 포함하는 SELECT 문
    - 부속 질의문(서브 질의문): 다른 SELECT 문 안에 들어 있는 SELECT 문
      - » 괄호로 묶어서 작성, ORDER BY 절을 사용할 수 없음
      - » 단일 행 부속 질의문: 하나의 행을 결과로 반환
      - » 다중 행 부속 질의문: 하나 이상의 행을 결과로 반환
  - 부속 질의문을 먼저 수행하고, 그 결과를 이용해 상위 질의문을 수행
  - 부속 질의문과 상위 질의문을 연결하는 연산자가 필요
    - 단일 행 부속 질의문은 비교 연산자(=, <>, >, >=, <, <=) 사용 가능
    - 다중 행 부속 질의문은 비교 연산자 사용 불가

03 SQL을 이용한 데이터 조작

❖ 데이터 검색 : SELECT 문

▪ 부속 질의문을 이용한 검색

예제 7-40

판매 데이터베이스에서 달콤비스킷을 생산한 제조업체가 만든 제품들의 제품명과 단가를 검색해보자.

```
SELECT  제품명, 단가
FROM    제품
WHERE   제조업체 = (SELECT  제조업체
                    FROM    제품
                    WHERE   제품명 = '달콤비스킷');
```

부속 질의문

상위 질의문

❶ ❷

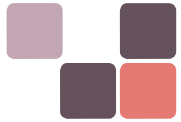
```
SELECT  제품명, 단가
FROM    제품
WHERE   제조업체 = '한빛제과';
```

결과 테이블

	제품명	단가
1	쿵떡파이	2600
2	맛난초콜릿	2500
3	달콤비스킷	1500

'달콤비스켓'의 제조업체는 '한빛제과'밖에 없기 때문에
단일 행 부속 질의문
(비교 연산자 =를 이용)

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 부속 질의문을 이용한 검색

예제 7-41

판매 데이터베이스에서 적립금이 가장 많은 고객의 고객이름과 적립금을 검색해보자.

```
▶▶ SELECT   고객이름, 적립금
   FROM     고객
   WHERE    적립금 = (SELECT  MAX(적립금)  FROM  고객);
```

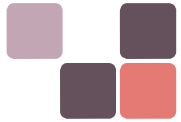
결과 테이블

	고객이름	적립금
1	성원용	5000

```
SELECT   고객이름, 적립금
FROM     고객
WHERE    적립금 = 5000;
```

최대 적립금은 단일 값이므로 단일 행 부속 질의문
(비교 연산자 =를 이용)

03 SQL을 이용한 데이터 조작



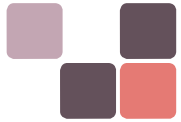
❖ 데이터 검색 : SELECT 문

- 부속 질의문을 이용한 검색

표 7-7 다중 행 부속 질의문에 사용 가능한 연산자

연산자	설명
IN	부속 질의문의 결과 값 중 일치하는 것이 있으면 검색 조건이 참
NOT IN	부속 질의문의 결과 값 중 일치하는 것이 없으면 검색 조건이 참
EXISTS	부속 질의문의 결과 값이 하나라도 존재하면 검색 조건이 참
NOT EXISTS	부속 질의문의 결과 값이 하나도 존재하지 않으면 검색 조건이 참
ALL	부속 질의문의 결과 값 모두와 비교한 결과가 참이면 검색 조건을 만족(비교 연산자와 함께 사용)
ANY 또는 SOME	부속 질의문의 결과 값 중 하나라도 비교한 결과가 참이면 검색 조건을 만족(비교 연산자와 함께 사용)

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 부속 질의문을 이용한 검색

예제 7-42

판매 데이터베이스에서 banana 고객이 주문한 제품의 제품명과 제조업체를 검색해보자.

```
SELECT    제품명, 제조업체
FROM      제품
WHERE     제품번호  IN (SELECT  주문제품
                      FROM    주문
                      WHERE   주문고객  = 'banana');
```

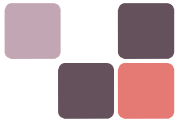
```
SELECT    제품명, 제조업체
FROM      제품
WHERE     제품번호 IN ('p01', 'p04', 'p06')
```

결과 테이블

	제품명	제조업체
1	그냥만두	대한식품
2	맛난초콜릿	한빛제과
3	통통우동	민국푸드

'banana' 고객이 주문한 제품은 여러 개이므로
다중 행 부속 질의문
(IN 연산자를 이용)

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 부속 질의문을 이용한 검색

예제 7-43

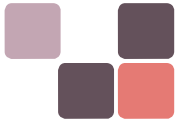
판매 데이터베이스에서 banana 고객이 주문하지 않은 제품의 제품명과 제조업체를 검색해 보자.

```
▶▶ SELECT   제품명, 제조업체
   FROM     제품
   WHERE    제품번호  NOT  IN (SELECT 주문제품
                          FROM   주문
                          WHERE  주문고객  = 'banana');
```

결과 테이블

	제품명	제조업체
1	쿵떡파이	한빛제과
2	매운쫄면	민국푸드
3	얼큰라면	대한식품
4	달콤비스킷	한빛제과

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 부속 질의문을 이용한 검색

예제 7-44

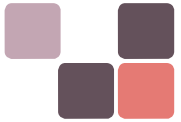
판매 데이터베이스에서 대한식품이 제조한 모든 제품의 단가보다 비싼 제품의 제품명, 단가, 제조업체를 검색해보자.

```
▶▶ SELECT    제품명, 단가, 제조업체
   FROM      제품
   WHERE     단가 > ALL ( SELECT  단가
                          FROM    제품
                          WHERE   제조업체 = '대한식품');
```

결과 테이블

	제품명	단가	제조업체
1	매운쫄면	5500	민국푸드

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 부속 질의문을 이용한 검색

예제 7-45

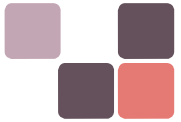
판매 데이터베이스에서 2019년 3월 15일에 제품을 주문한 고객의 고객이름을 검색해보자.

```
▶▶ SELECT    고객이름
   FROM      고객
   WHERE     EXISTS (SELECT  *
                     FROM    주문
                     WHERE   주문일자 = '2019-03-15'
                             AND  주문.주문고객 = 고객.고객아이디);
```

결과 테이블

	고객이름
1	정소화

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

▪ 부속 질의문을 이용한 검색

예제 7-46

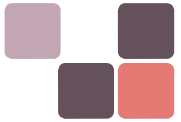
판매 데이터베이스에서 2019년 3월 15일에 제품을 주문하지 않은 고객의 고객이름을 검색해보자.

```
▶▶ ELECT     고객이름
   FROM      고객
   WHERE     NOT  EXISTS (SELECT  *
                          FROM   주문
                          WHERE  주문일자 = '2019-03-15'
                                 AND  주문.주문고객 = 고객.고객아이디);
```

결과 테이블

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

03 SQL을 이용한 데이터 조작



❖ 데이터 검색 : SELECT 문

- 질의 내용은 다양하게 표현 가능하므로 사용자가 자유롭게 선택

❶ 조인 질의를 이용한 SELECT 문

```
SELECT  제품.제품명, 제품.제조업체
FROM    제품, 주문
WHERE   제품.제품번호 = 주문.주문제품  AND  주문.주문고객 = 'banana';
```

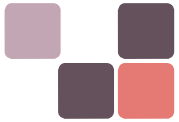
❷ EXISTS 연산자를 이용한 SELECT 문

```
SELECT 제품명, 제조업체
FROM   제품
WHERE  EXISTS (SELECT  *
               FROM    주문
               WHERE   제품.제품번호 = 주문.주문제품
                       AND  주문.주문고객 = 'banana');
```

[예제 7-42]

```
SELECT 제품명, 제조업체
FROM   제품
WHERE  제품번호 IN (SELECT 주문제품
                    FROM   주문
                    WHERE  주문고객 = 'banana');
```

03 SQL을 이용한 데이터 조작



❖ 데이터 삽입 : INSERT 문

- 데이터 직접 삽입

```
INSERT
INTO    테이블_이름[(속성_리스트)]
VALUES (속성값_리스트);
```

- INTO 키워드와 함께 투플을 삽입할 테이블의 이름과 속성의 이름을 나열
  - 속성 리스트를 생략하면 테이블을 정의할 때 지정한 속성의 순서대로 값이 삽입됨
- VALUES 키워드와 함께 삽입할 속성 값들을 나열
- INTO 절의 속성 이름과 VALUES 절의 속성 값은 순서대로 일대일 대응되어야 함

03 SQL을 이용한 데이터 조작

데이터 삽입 : INSERT 문

데이터 직접 삽입

예제 7-47

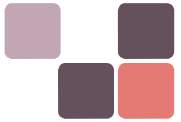
판매 데이터베이스의 고객 테이블에 고객아이디가 strawberry, 고객이름이 최유경, 나이가 30세, 등급이 vip, 직업이 공무원, 적립금이 100원인 새로운 고객의 정보를 삽입해보자. 그런 다음 고객 테이블에 있는 모든 내용을 검색하여 삽입된 새로운 투플을 확인해보자.

```
INSERT
INTO      고객(고객아이디, 고객이름, 나이, 등급, 직업, 적립금)
VALUES    ('strawberry', '최유경', 30, 'vip', '공무원', 100);

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
8	strawberry	최유경	30	vip	공무원	100

03 SQL을 이용한 데이터 조작



❖ 데이터 삽입 : INSERT 문

- 데이터 직접 삽입

```
INSERT
INTO    고객(고객아이디, 고객이름, 나이, 등급, 직업, 적립금)
VALUES  ('strawberry', '최유경', 30, 'vip', '공무원', 100);
```

=

```
INSERT
INTO    고객
VALUES  ('strawberry', '최유경', 30, 'vip', '공무원', 100);
```

03 SQL을 이용한 데이터 조작

데이터 삽입 : INSERT 문

데이터 직접 삽입

예제 7-48

판매 데이터베이스의 고객 테이블에 고객아이디가 tomato, 고객이름이 정은심, 나이가 36세, 등급이 gold, 적립금은 4,000원, 직업은 아직 모르는 새로운 고객의 정보를 삽입해보자. 그런 다음 고객 테이블에 있는 모든 내용을 검색하여, 삽입된 정은심 고객의 직업 속성이 널 값인지 확인해보자.

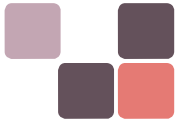
```
INSERT
INTO      고객(고객아이디, 고객이름, 나이, 등급, 적립금)
VALUES    ('tomato', '정은심', 36, 'gold', 4000);

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
8	strawberry	최유경	30	vip	공무원	100
9	tomato	정은심	36	gold	(null)	4000

직업 속성에 널 값을 삽입

03 SQL을 이용한 데이터 조작



❖ 데이터 삽입 : INSERT 문

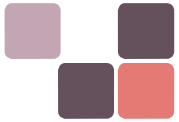
- 데이터 직접 삽입

```
INSERT
INTO    고객(고객아이디, 고객이름, 나이, 등급, 적립금)
VALUES  ('tomato', '정은심', 36, 'gold', 4000);
```

=

```
INSERT
INTO    고객
VALUES  ('tomato', '정은심', 36, 'gold', NULL, 4000);
```

03 SQL을 이용한 데이터 조작



❖ 데이터 삽입 : INSERT 문

- 부속 질의문을 이용한 데이터 삽입
  - SELECT 문을 이용해 다른 테이블에서 검색한 데이터를 삽입

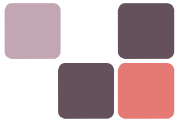
```
INSERT
INTO  테이블_이름[(속성_리스트)]
SELECT 문;
```

예 INSERT

```
INTO      한빛제품(제품명, 재고량, 단가)
SELECT    제품명, 재고량, 단가
FROM      제품
WHERE     제조업체 = '한빛제과';
```

한빛제과에서 제조한 제품의 제품명, 재고량, 단가를 제품 테이블에서 검색하여 한빛제품 테이블에 삽입

03 SQL을 이용한 데이터 조작



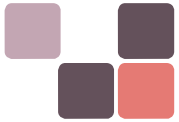
❖ 데이터 수정 : UPDATE 문

- 테이블에 저장된 투플에서 특정 속성의 값을 수정

```
UPDATE  테이블_이름
SET     속성_이름1 = 값1, 속성_이름2 = 값2, …
[WHERE  조건];
```

- SET 키워드 다음에 속성 값을 어떻게 수정할 것인지를 지정
- WHERE 절에 제시된 조건을 만족하는 투플에 대해서만 속성 값을 수정
  - WHERE 절을 생략하면 테이블에 존재하는 모든 투플을 대상으로 수정

03 SQL을 이용한 데이터 조작



❖ 데이터 수정 : UPDATE 문

예제 7-49

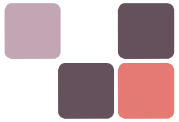
제품 테이블에서 제품번호가 p03인 제품의 제품명을 통큰파이로 수정해보자.

```
▶▶ UPDATE    제품
   SET       제품명 = '통큰파이'
   WHERE     제품번호 = 'p03';

   SELECT * FROM 제품;
```

결과 테이블

	제품번호	제품명	재고량	단가	제조업체
1	p01	그냥만두	5000	4500	대한식품
2	p02	매운쫄면	2500	5500	민국푸드
3	p03	통큰파이	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

03 SQL을 이용한 데이터 조작



❖ 데이터 수정 : UPDATE 문

예제 7-50

제품 테이블에 있는 모든 제품의 단가를 10% 인상해보자. 그런 다음 제품 테이블의 모든 내용을 검색하여 인상 내용을 확인해보자.

```
UPDATE    제품
SET       단가 = 단가 * 1.1;

SELECT * FROM 제품;
```

결과 테이블

	제품번호	제품명	재고량	단가	제조업체
1	p01	그냥만두	5000	4950	대한식품
2	p02	매운쫄면	2500	6050	민국푸드
3	p03	통큰파이	3600	2860	한빛제과
4	p04	맛난초콜릿	1250	2750	한빛제과
5	p05	얼큰라면	2200	1320	대한식품
6	p06	통통우동	1000	1705	민국푸드
7	p07	달콤비스킷	1650	1650	한빛제과

03 SQL을 이용한 데이터 조작

❖ 데이터 수정 : UPDATE 문

예제 7-51

판매 데이터베이스에서 정소화 고객이 주문한 제품의 주문수량을 5개로 수정해보자. 그런 다음 주문 테이블의 모든 내용을 검색하여 수정 내용을 확인해보자.

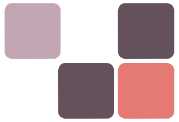
```
▶▶ UPDATE    주문
   SET       수량 = 5
   WHERE     주문고객 IN (SELECT  고객아이디
                          FROM    고객
                          WHERE   고객이름 = '정소화');

   SELECT * FROM 주문;
```

부속 질의문을 이용한 UPDATE 문

결과 테이블

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

03 SQL을 이용한 데이터 조작



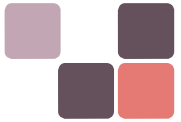
❖ 데이터 삭제 : DELETE 문

- 테이블에 저장된 데이터를 삭제

```
DELETE
FROM   테이블_이름
[WHERE 조건];
```

- WHERE 절에 제시한 조건을 만족하는 투플만 삭제
  - WHERE 절을 생략하면 테이블에 존재하는 모든 투플을 삭제해 빈 테이블이 됨

03 SQL을 이용한 데이터 조작



데이터 삭제 : DELETE 문

예제 7-52

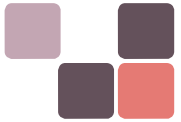
주문 테이블에서 주문일자가 2019년 5월 22일인 주문내역을 삭제해보자. 그런 다음 주문 테이블의 모든 내용을 검색하여 삭제 여부를 확인해보자.

```
DELETE
FROM      주문
WHERE     주문일자 = '2019-05-22';

SELECT * FROM 주문;
```

결과 테이블

	주문번호	주문고객	주문제품	수량	배송지	주문일자
1	o01	apple	p03	5	서울시 마포구	19/01/01
2	o02	melon	p01	5	인천시 계양구	19/01/10
3	o03	banana	p06	45	경기도 부천시	19/01/11
4	o04	carrot	p02	8	부산시 금정구	19/02/01
5	o05	melon	p06	36	경기도 용인시	19/02/20
6	o06	banana	p01	19	충청북도 보은군	19/03/02
7	o07	apple	p03	5	서울시 영등포구	19/03/15
8	o08	pear	p02	50	강원도 춘천시	19/04/10
9	o09	banana	p04	15	전라남도 목포시	19/04/11

03 SQL을 이용한 데이터 조작



❖ 데이터 삭제 : DELETE 문

예제 7-53

판매 데이터베이스에서 정소화 고객이 주문한 내역을 주문 테이블에서 삭제해보자. 그런 다음 주문 테이블의 모든 내용을 검색하여 삭제 여부를 확인해보자.

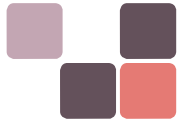
```
▶▶ DELETE
   FROM     주문
   WHERE    주문고객 IN (SELECT   고객아이디
                         FROM     고객
                         WHERE    고객이름 = '정소화');

   SELECT * FROM 주문;
```

부속 질의문을 이용한 DELETE 문

결과 테이블

	주문번호	주문고객	주문제품	수량	배송지	주문일자
1	o02	melon	p01	5	인천시 계양구	19/01/10
2	o03	banana	p06	45	경기도 부천시	19/01/11
3	o04	carrot	p02	8	부산시 금정구	19/02/01
4	o05	melon	p06	36	경기도 용인시	19/02/20
5	o06	banana	p01	19	충청북도 보은군	19/03/02
6	o08	pear	p02	50	강원도 춘천시	19/04/10
7	o09	banana	p04	15	전라남도 목포시	19/04/11

03 SQL을 이용한 데이터 조작



데이터 삭제 : DELETE 문

예제 7-54

판매 데이터베이스의 주문 테이블에 존재하는 모든 투플을 삭제해보자. 그런 다음 주문 테이블의 모든 내용을 검색하여 삭제 여부를 확인해보자.

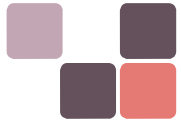
```
DELETE
FROM      주문;

SELECT * FROM 주문;
```

빈 테이블이 남음, DROP TABLE과는 다름

결과 테이블

주문번호	주문고객	주문제품	수량	배송지	주문일자

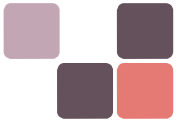
04 뷰



❖ 뷰(View)

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

04 뷰



- 뷰는 기본 테이블을 들여다 볼 수 있는 창의 역할을 담당

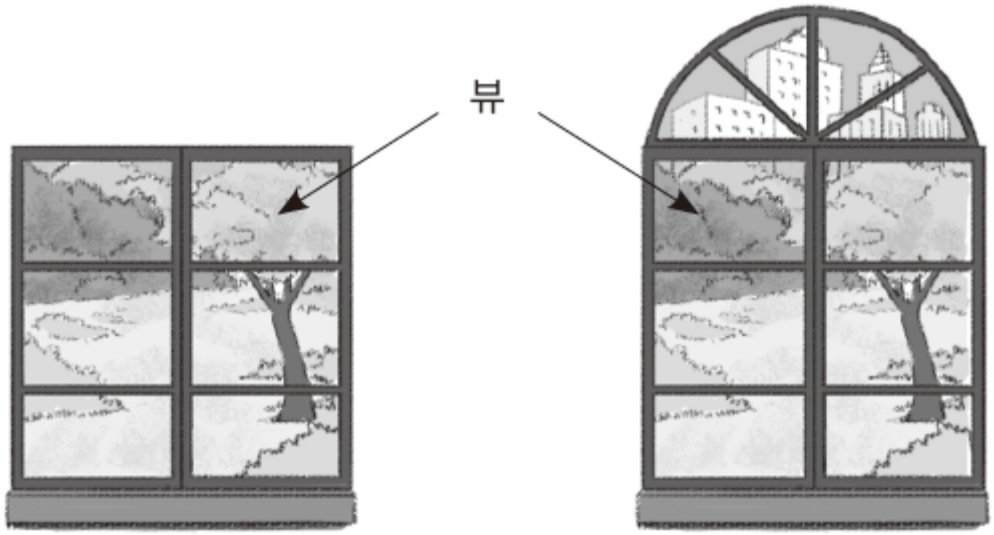
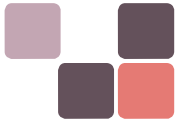


그림 7-12 뷰의 창 역할

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 뷰를 구성하는 속성의 이름을 나열
  - 속성 리스트를 생략하면 SELECT 절에 나열된 속성의 이름을 그대로 사용
- AS 키워드와 함께 기본 테이블에 대한 SELECT 문 작성
  - SELECT 문은 생성하려는 뷰의 정의를 표현하며 ORDER BY는 사용 불가
- 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

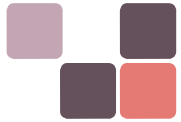
고객 테이블에서 등급이 vip인 고객의 고객아이디, 고객이름, 나이로 구성된 뷰를 우수고객이라는 이름으로 생성해보자. 그런 다음 우수고객 뷰의 모든 내용을 검색해보자.

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

SELECT  *  FROM  우수고객;
```

결과 테이블

	고객아이디	고객이름	나이
1	banana	김선우	25

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)

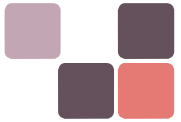


❖ 뷰 생성 : CREATE VIEW 문

```
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;
```



❖ 뷰 생성 : CREATE VIEW 문

예제 7-56

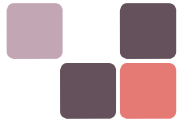
제품 테이블에서 제조업체별 제품수로 구성된 뷰를 업체별제품수라는 이름으로 생성해보자. 그런 다음 업체별제품수 뷰의 모든 내용을 검색해보자.

```
▶▶ 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



❖ 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

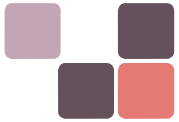
우수고객 뷰에서 나이가 25세 이상인 고객에 대한 모든 내용을 검색해보자.

```
▶▶ SELECT * FROM 우수고객 WHERE 나이 >= 25;
```

결과 테이블

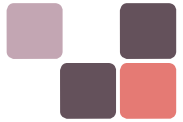
	고객아이디	고객이름	나이
1	banana	김선우	25

04 뷰



❖ 뷰 활용 : INSERT, UPDATE, DELETE 문

- 뷰에 대한 삽입·수정·삭제 연산은 실제로 기본 테이블에 수행되므로 결과적으로는 기본 테이블이 변경됨
- 뷰에 대한 삽입·수정·삭제 연산은 제한적으로 수행됨
  - 변경 가능한 뷰 vs. 변경 불가능한 뷰
- 변경 불가능한 뷰의 특징
  - 기본 테이블의 기본키를 구성하는 속성이 포함되어 있지 않은 뷰
  - 기본 테이블에 있던 내용이 아닌 집계 함수로 새로 계산된 내용을 포함하는 뷰
  - DISTICNT 키워드를 포함하여 정의한 뷰
  - GROUP BY 절을 포함하여 정의한 뷰
  - 여러 개의 테이블을 조인하여 정의한 뷰는 변경이 불가능한 경우가 많음

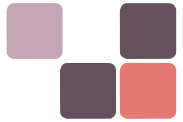


❖ 뷰 활용 : INSERT, UPDATE, DELETE 문

```
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

제품1 뷰는 변경 가능한 뷰인가?

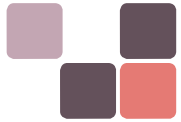


❖ 뷰 활용 : INSERT, UPDATE, DELETE 문

```
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?



❖ 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

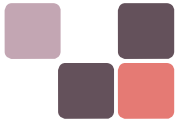
04 뷰

❖ 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.



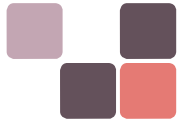
❖ 뷰 활용 : INSERT, UPDATE, DELETE 문

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

제품2 뷰에 대한 삽입 연산은 실패함(오류 발생)

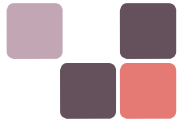
➔ 제품2 뷰는 제품 테이블의 기본키인 제품번호 속성을 포함하고 있지 않기 때문에
제품2 뷰를 통해 새로운 투플을 삽입하려고 하면
제품번호 속성이 널 값이 되어 삽입 연산에 실패하게 됨

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

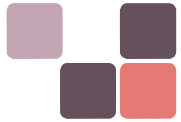


❖ 뷰 삭제 : DROP VIEW 문

- 뷰를 삭제해도 기본 테이블은 영향을 받지 않음

```
DROP  VIEW  뷰_이름;
```

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



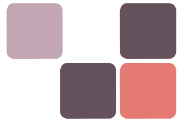
❖ 뷰 삭제 : DROP VIEW 문

예제 7-59

우수고객 뷰를 삭제해보자.

```
▶▶ DROP  VIEW  우수고객;
```

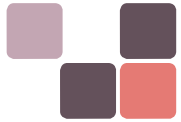
05 삽입 SQL



❖ 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

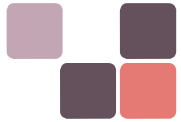
05 삽입 SQL

삽입 SQL 문은 "EXEC SQL"을 붙여서 일반 명령문과 구분함

```
int main() {

    EXEC  SQL  BEGIN  DECLARE  SECTION;
❶         char   p_no[4],  p_name[21];
          int    price;
    EXEC  SQL  END  DECLARE  SECTION;

❷   printf("제품번호를 입력하세요 : ");
    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);
    printf("\n 단가 = %d", price);

    return 0;
}
```

그림 7-14 입력된 제품번호에 해당되는 제품명과 단가를 검색하는 프로그램

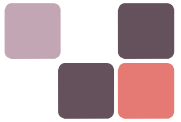
05 삽입 SQL



❖ [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 문을 선언

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

예 EXEC SQL DECLARE product_cursor CURSOR FOR
SELECT 제품명, 단가 FROM 제품;

제품 테이블에서 제품명과 단가를 모두 검색하는 SELECT 문을 위한 커서를 product_cursor라는 이름으로 선언

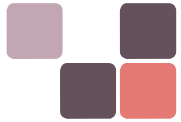
- 커서에 연결된 SELECT 문을 실행하는 삽입 SQL 문

```
EXEC SQL OPEN 커서_이름;
```

예 EXEC SQL OPEN product_cursor;

product_cursor라는 이름의 커서에 연결된 SELECT 문 실행

05 삽입 SQL



❖ 커서가 필요한 삽입 SQL

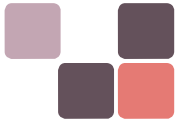
- 커서를 이동시키는 삽입 SQL 문
  - 커서를 이동하여 처리할 다음 행을 가리키도록 하고, 커서가 가리키는 행으로부터 속성 값을 가져와 변수에 저장시킴
  - 결과 테이블에는 여러 행이 존재하므로 FETCH 문은 여러 번 수행해야 함
    - for, while 문과 같은 반복문과 함께 사용

```
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 저장

05 삽입 SQL

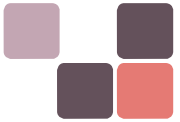


❖ 커서가 필요한 삽입 SQL

- 커서의 사용을 종료하는 삽입 SQL 문

```
EXEC  SQL  CLOSE  커서_이름;
```

예 EXEC SQL CLOSE procuct_cursor; product_cursor 커서를 더는 사용하지 않음



Thank You