제1장 SQL 기초

1) 데이터베이스 생성 및 제거

데이터베이스 생성

```
CREATE DATABASE 데이터베이스명;
```

데이터베이스 제거

```
DROP DATABASE 데이터베이스명;
```

☞ 실습 1-1

```
> CREATE DATABASE `UserDB`;
> DROP DATABASE `UserDB`;
```

2) 테이블 생성 및 제거

테이블 생성

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

테이블 제거

```
DROP TABLE 테이블명;
DROP TABLE IF EXISTS 테이블명;
```

☞ 실습 1-2

```
> CREATE TABLE `User1` (
    `uid` VARCHAR(10),
    `name` VARCHAR(10),
    `hp` CHAR(13),
    `age` INT
    );

> DROP TABLE `User1`;
```

3) 테이블에 데이터 추가

데이터 추가

```
INSERT INTO 테이블명 VALUES (데이터1, 데이터2, 데이터3 ...);
```

칼럼명을 지정해서 데이터 추가

```
INSERT INTO 테이블명 (칼럼명1, 칼럼명2 ... ) VALUES (데이터1, 데이터2 ...);
```

칼럼명을 지정해서 데이터 추가

```
INSERT INTO 테이블명 SET 칼럼명1=데이터1, 칼럼명2=데이터2 ...;
```

☞ 실습 1-3

```
> INSERT INTO `User1` VALUES ('A101', '김유신', '010-1234-1111', 25);
> INSERT INTO `User1` VALUES ('A102', '김춘추', '010-1234-2222', 23);
> INSERT INTO `User1` VALUES ('A103', '장보고', '010-1234-3333', 32);
> INSERT INTO `User1` (`uid`, `name`, `age`) VALUES ('A104', '강감찬', 45);
> INSERT INTO `User1` SET `uid`='A105', `name`='이순신', `hp`='010-1234-5555';
```

4) 테이블에 데이터 조회

데이터 조회

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

모든 데이터 조회

```
SELECT * FROM 테이블명;
SELECT * FROM 데이터베이스명.테이블명;
```

☞ 실습 1-4

```
> SELECT * FROM `User1`;
> SELECT * FROM `User1` WHERE `uid`='A101';
> SELECT * FROM `User1` WHERE `name`='김춘추';
> SELECT * FROM `User1` WHERE `age` < 30;
> SELECT * FROM `User1` WHERE `age` >= 30;
> SELECT `uid`, `name`, `age` FROM `User1`;
```

5) 테이블에 데이터 수정

데이터 수정

```
UPDATE 테이블명 SET 칼럼명1=데이터1, 칼럼명2=데이터2 ...;
```

조건에 일치하는 레코드만 수정

```
UPDATE 테이블명 SET 칼럼명1=데이터1, 칼럼명2=데이터2 ... WHERE 조건;
```

☞ 실습 1-5

```
> UPDATE `User1` SET `hp`='010-1234-4444' WHERE `uid`='A104';
> UPDATE `User1` SET `age`=51 WHERE `uid`='A105';
> UPDATE `User1` SET `hp`='010-1234-1001', `age`=27 WHERE `uid`='A101';
```

6) 테이블에 데이터 삭제

데이터 삭제

```
DELETE FROM 테이블명 WHERE 조건;
```

☞ 실습 1-6

```
> DELETE FROM `User1` WHERE `uid`='A101';
> DELETE FROM `User1` WHERE `uid`='A102' AND `age`=25;
> DELETE FROM `User1` WHERE `age` >= 30;
```

7) 연습문제

☞ 실습 1-7

아래 SQL을 생성하시오.

· 회원 테이블(TblUser)

```
CREATE TABLE `TblUser` (
        `userId`        VARCHAR(10),
        `userName`      VARCHAR(10),
        `userHp`        CHAR(13),
        `userAge`       TINYINT,
        `userAddr`      VARCHAR(20)
);
```

· 제품 테이블(TblProduct)

```
CREATE TABLE `TblProduct` (
        `prdCode`       INT,
        `prdName`       VARCHAR(10),
        `prdPrice`      INT,
        `prdAmount`     INT,
        `prdCompany`    VARCHAR(10),
        `prdMakeDate`   DATE
);
```

☞ 실습 1-8

테이블에 아래 그림과 같이 데이터를 입력하시오.

· 회원 테이블(TblUser)

userId	userName	userHp	userAge	userAddr
p101	김유신	010-1234-1001	25	서울시 종구
p102	김춘추	010-1234-1002	23	부산시 금정구
p103	장보고	(NULL)	31	경기도 광주군
p104	강감찬	(NULL)	(NULL)	경남 창원시
p105	이순신	010-1234-1005	50	(NULL)

· 제품 테이블(TblProduct)

prdCode	prdName	prdPrice	prdAmount	prdCompany	prdMakeDate
1	냉장고	800	10	LG	2022-01-06
2	노트북	1,200	20	삼성	2022-01-06
3	TV	1,400	6	LG	2022-01-06
4	세탁기	1,000	8	LG	2022-01-06
5	컴퓨터	1,100	0	(NULL)	(NULL)
6	휴대폰	900	102	삼성	2022-01-06

☞ 실습 1-9

아래 SQL을 실행하시오.

```
> SELECT * FROM `TblUser`;
> SELECT `userName` FROM `TblUser`;
> SELECT `userName`, `userHp` FROM `TblUser`;
> SELECT * FROM `TblUser` WHERE `userId`='p102';
> SELECT * FROM `TblUser` WHERE `userId`='p104' OR `userId`='p105';
> SELECT * FROM `TblUser` WHERE `userAddr`='부산시 금정구';
> SELECT * FROM `TblUser` WHERE `userAge` > 30;
> SELECT * FROM `TblUser` WHERE `userHP` IS NULL;
> UPDATE `TblUser` SET `userAge`=42 WHERE `userID`='p104';
> UPDATE `TblUser` SET `userAddr`='경남 김해시' WHERE `userId`='p105';
> DELETE FROM `TblUser` WHERE `userId`='p103';

> SELECT * FROM `TblProduct`;
> SELECT `prdName` FROM `TblProduct`;
> SELECT `prdName`, `prdPrice` FROM `TblProduct`;
> SELECT * FROM `TblProduct` WHERE `prdCompany`='LG';
> SELECT * FROM `TblProduct` WHERE `prdCompany`='삼성';
> UPDATE `TblProduct` SET
                        `prdCompany`='삼성',
                        `prdMakeDate`='2021-01-01'
                      WHERE
                        `prdCode`=5;
```

제2장 테이블 제약조건과 수정

1) 기본키(Primary Key)

① 제약조건(Constraint)이란 데이터의 무결성을 지키기 위한 제한된 조건을 의미

② 기본 키(PK)는 테이블에 존재하는 많은 행의 데이터를 구분할 수 있는 식별값 역할

테이블을 생성할 때 기본키 지정하기

```
CREATE TABLE 테이블명 (칼럼명 자료형 PRIMARY KEY...);
```

※ PK 칼럼의 특징

- 값이 중복되지 않는다.
- 반드시 데이터를 입력해야 한다.

☞ 실습 2-1

```
> CREATE TABLE `User2` (
    `uid`   VARCHAR(10) PRIMARY KEY,
    `name`  VARCHAR(10),
    `hp`    CHAR(13),
    `age`   INT
    );
```

☞ 실습 2-2. User2 데이터 추가하기

2) 고유키(Unique)

① 고유 키(UNIQUE) 제약조건은 '중복되지 않은 유일한 값'을 입력해야 하는 조건

② 기본 키와 비슷하며 차이점은 UNIQUE는 NULL 값을 허용

테이블을 생성할 때 고유키 지정하기

```
CREATE TABLE 테이블명 (칼럼명 자료형 UNIQUE...);
```

※ UNIQUE 칼럼의 특징

- 값이 중복되지 않는다.
- NULL값을 입력 할 수 있다.

☞ 실습 2-3

```
> CREATE TABLE `User3` (
    `uid`   VARCHAR(10) PRIMARY KEY,
    `name`  VARCHAR(10),
    `hp`    CHAR(13) UNIQUE,
    `age`   INT
    );
```

☞ 실습 2-4. User3 데이터 추가하기

3) 외래키(Foreign Key)

① 외래키는 두 테이블을 서로 연결하는 데 사용되는 키

② 외래키를 가진 테이블이 자식 테이블이라고 하고 외래키 값을 제공하는 테이블을 부모 테이블

③ 외래키 값은 NULL이거나 부모 테이블의 기본키 값과 동일(참조 무결성 제약조건)

테이블 외래키 지정하기

```
CREATE TABLE 테이블명 (칼럼명 자료형, FOREIGN KEY (`column1`) REFERENCES `Parent` (`id1`));
```

☞ 실습 2-5

```
> CREATE TABLE `Parent` (
    `pid`   VARCHAR(10) PRIMARY KEY,
    `name`  VARCHAR(10),
    `hp`    CHAR(13) UNIQUE
    );

> CREATE TABLE `Child` (
    `cid`    VARCHAR(10) PRIMARY KEY,
    `name`   VARCHAR(10),
    `hp`     CHAR(13) UNIQUE,
    `pid`    VARCHAR(10),
    FOREIGN KEY (`pid`) REFERENCES `Parent` (`pid`)
   );
```

☞ 실습 2-6. Parent, Child 데이터 추가하기

4) 자동번호 컬럼
① 시퀀스 칼럼은 레코드가 추가되면 1부터 1씩 증가하면서 숫자가 자동으로 입력되는 칼럼
② INT형 칼럼에 AUTO_INCREMENT 키워드를 지정해서 시퀀스 칼럼을 설정
③ 기본 키를 지정할 후보 키가 없는 테이블에 사용

테이블을 생성할 때 시퀀스 칼럼 지정

```
CREATE TABLE 테이블명 (칼럼명 자료형 PRIMARY KEY AUTO_INCREMENT ... );
```

☞ 실습 2-7

```
> CREATE TABLE `User4`(
    `seq`    INT AUTO_INCREMENT PRIMARY KEY,
    `name`   VARCHAR(10),
    `gender`TINYINT,
    `age`    INT,
    `addr`   VARCHAR(255)
    );
```

☞ 실습 2-8. User4 데이터 추가하기

5) DEFAULT와 NULL값
① DEFAULT는 값을 입력하지 않아도 자동으로 입력되는 기본값
② 모든 컬럼의 DEFAULT는 NULL이고, NULL은 아직 아무 것도 없는 값을 의미
③ 반드시 데이터 입력을 받아야 하는 컬럼은 NOT NULL을 선언

☞ 실습 2-9

```
> CREATE TABLE `User5`(
    `name`   VARCHAR(10) NOT NULL,
    `gender`TINYINT,
    `age`    INT DEFAULT 1,
    `addr`   VARCHAR(10)
    );
```

☞ 실습 2-10. User5 데이터 추가하기

6) 테이블 수정

속성(열) 추가

```
ALTER TABLE 테이블명 ADD 속성명 자료형;
```

☞ 실습 2-11

```
> ALTER TABLE `User5` ADD `hp` VARCHAR(20);
> ALTER TABLE `User5` ADD `birth` CHAR(10) DEFAULT '0000-00-00' AFTER `name`;
```

속성(열) 자료형 변경

```
ALTER TABLE 테이블명 MODIFY 속성명 새로운_자료형;
```

☞ 실습 2-12

```
> ALTER TABLE `User5` MODIFY `hp` CHAR(13);
> ALTER TABLE `User5` MODIFY `age` TINYINT;
```

속성(열) 이름과 자료형 변경

```
ALTER TABLE 테이블명 CHANGE COLUMN 속성명 새로운_속성명 자료형;
```

☞ 실습 2-13

```
> ALTER TABLE `User5` CHANGE COLUMN `addr` `address` VARCHAR(100);
```

속성(열) 삭제

```
ALTER TABLE 테이블명 DROP COLUMN 속성명;
```

☞ 실습 2-14

```
> ALTER TABLE `User5` DROP `gender`;
```

7) 테이블 복사

테이블 복사

```
CREATE TABLE 새로운_테이블명 LIKE 복사할_테이블명;
```

※ PK, UNIQUE, AUTO_INCREMENT 등의 속성도 같이 복사

☞ 실습 2-15

```
> CREATE TABLE `User6` LIKE `User5`;
```

테이블 데이터 복사

```
INSERT INTO 테이블명 SELECT * FROM 복사할_테이블명;
INSERT INTO 테이블명 (칼럼명...) SELECT 칼럼명... FROM 복사할_테이블명;
```

☞ 실습 2-16

```
> INSERT INTO `User6` SELECT * FROM `User5`;
```

제3장 데이터베이스 관리

1) 관리자 추가

일반 관리자 생성

```
CREATE USER '아이디'@'%' IDENTIFIED BY '비밀번호';
```

권한부여

```
GRANT ALL PRIVILEGES ON 데이터베이스.* TO '아이디'@'%';
```

변경사항 적용

```
FLUSH PRIVILEGES;
```

[참고] %는 모든 외부 IP 접속을 허용

☞ 실습 3-1

```
> CREATE DATABASE `TestDB`;
> CREATE USER 'tester'@'%' IDENTIFIED BY '1234';
> GRANT ALL PRIVILEGES ON TestDB.* TO 'tester'@'%';
> FLUSH PRIVILEGES;
```

2) 관리자 비밀번호 수정 및 계정 삭제

비밀번호 변경

```
SET PASSWORD FOR '아이디'@'%'=PASSWORD('새로운 비밀번호');
```

계정 삭제

```
DROP USER '아이디'@'%';
```

☞ 실습 3-2

```
> alter user 'tester'@'%' identified by '12345';
> drop USER 'tester'@'%';
> FLUSH PRIVILEGES;
```

제4장 SQL 고급

1) SQL 실습 준비하기

☞ 실습 4-1

아래 표를 참고하여 테이블을 생성하시오.

· 직원 테이블(Member)

컬럼명(한글)	영문명	데이터유형	길이	NULL허용	기본값
아이디(PK)	uid	문자열	10	X	없음
이름	name	문자열	10	X	없음
휴대폰(UK)	hp	문자열	13	X	없음
직급	pos	문자열	10	X	사원
부서번호	dep	숫자	정수	O	NULL
입사일	rdate	날짜시간	날짜시간	X	없음

· 부서 테이블(Department)

컬럼명(한글)	영문명	데이터유형	길이	NULL허용	기본값
부서번호(PK)	depNo	숫자	정수	X	없음
부서명	name	문자열	10	X	없음
부서전화번호	tel	문자열	12	X	없음

· 매출 테이블(Sales)

컬럼명(한글)	영문명	데이터유형	길이	NULL허용	기본값
번호(PK)	seq	숫자	정수	X	AUTO_INCREMENT
직원아이디	uid	문자열	10	X	없음
매출연도	year	년도	년도	X	없음
매출월	month	숫자	정수	X	없음
매출액	sale	숫자	정수	X	없음

☞ 실습 4-2

테이블에 아래 그림과 같이 데이터를 입력하시오.

· 직원 테이블(Member)

uid	name	hp	pos	dep	rdate
a101	박혁거세	010-1234-1001	부장	101	2020-11-19 11:39:48
a102	김유신	010-1234-1002	차장	101	2020-11-19 11:39:48
a103	김춘추	010-1234-1003	사원	101	2020-11-19 11:39:48
a104	장보고	010-1234-1004	대리	102	2020-11-19 11:39:48
a105	강감찬	010-1234-1005	과장	102	2020-11-19 11:39:48
a106	이성계	010-1234-1006	차장	103	2020-11-19 11:39:48
a107	정철	010-1234-1007	차장	103	2020-11-19 11:39:48
a108	이순신	010-1234-1008	부장	104	2020-11-19 11:39:48
a109	허균	010-1234-1009	부장	104	2020-11-19 11:39:48
a110	정약용	010-1234-1010	사원	105	2020-11-19 11:39:48
a111	박지원	010-1234-1011	사원	105	2020-11-19 11:39:48

· 매출 테이블(Sales)

seq	uid	year	month	sale
1	a101	2018	1	98,100
2	a102	2018	1	136,000
3	a103	2018	1	80,100
4	a104	2018	1	78,000
5	a105	2018	1	93,000
6	a101	2018	2	23,500
7	a102	2018	2	126,000
8	a103	2018	2	18,500
9	a105	2018	2	19,000
10	a106	2018	2	53,000
11	a101	2019	1	24,000
12	a102	2019	1	109,000
13	a103	2019	1	101,000
14	a104	2019	1	53,500
15	a107	2019	1	24,000
16	a102	2019	2	160,000
17	a103	2019	2	101,000
18	a104	2019	2	43,000
19	a105	2019	2	24,000
20	a106	2019	2	109,000
21	a102	2020	1	201,000
22	a104	2020	1	63,000
23	a105	2020	1	74,000
24	a106	2020	1	122,000
25	a107	2020	1	111,000
26	a102	2020	2	120,000
27	a103	2020	2	93,000
28	a104	2020	2	84,000
29	a105	2020	2	180,000
30	a108	2020	2	76,000

· 부서 테이블(Department)

depNo	name	tel
101	영업1부	051-512-1001
102	영업2부	051-512-1002
103	영업3부	051-512-1003
104	영업4부	051-512-1004
105	영업5부	051-512-1005
106	영업지원부	051-512-1006
107	인사부	051-512-1007

2) 다양한 조건으로 데이터 조회하기

구분	연산자	설명
비교연산자	=	같다
	>	보다 크다
	>=	보다 크거나 같다
	<	보다 작다
	<=	보다 작거나 같다
	<>, !=	다르다
논리연산자	AND	그리고
	OR	또는
	A IN(B)	B 중에 A가 있다
	A NOT IN(B)	B 중에 A가 없다
	A BETWEEN ~ AND ...	~부터 ... 사이에 A가 있다
	A NOT BETWEEN ~ AND ...	~부터 ... 사이에 A가 없다
검색연산자	LIKE	와일드카드(%, _)로 문자열을 검색

조건에 일치하는 레코드 조회

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

☞ 실습 4-3

```
> SELECT * FROM `Member` WHERE `name`='김유신';
> SELECT * FROM `Member` WHERE `pos`='차장' AND dep=101;
> SELECT * FROM `Member` WHERE `pos`='차장' OR  dep=101;
> SELECT * FROM `Member` WHERE `name` != '김춘추';
> SELECT * FROM `Member` WHERE `name` <> '김춘추';
> SELECT * FROM `Member` WHERE `pos`='사원' OR `pos`='대리';
> SELECT * FROM `Member` WHERE `pos` IN('사원', '대리');
> SELECT * FROM `Member` WHERE `dep` IN(101, 102, 103);
> SELECT * FROM `Member` WHERE `name` LIKE '%신';
> SELECT * FROM `Member` WHERE `name` LIKE '김%';
> SELECT * FROM `Member` WHERE `name` LIKE '김_ _';
> SELECT * FROM `Member` WHERE `name` LIKE '_성_';
> SELECT * FROM `Member` WHERE `name` LIKE '정_';
> SELECT * FROM `Sales` WHERE `sale` > 50000;
> SELECT * FROM `Sales` WHERE `sale` >= 50000 AND `sale` < 100000 AND `month`=1;
> SELECT * FROM `Sales` WHERE `sale` BETWEEN 50000 AND 100000;
> SELECT * FROM `Sales` WHERE `sale` NOT BETWEEN 50000 AND 100000;
> SELECT * FROM `Sales` WHERE `year` IN(2020);
> SELECT * FROM `Sales` WHERE `month` IN(1, 2);
```

오름/내림차순으로 정렬

```
SELECT 칼럼명1, 칼럼명2 ... FROM 테이블명 ORDER BY 기준_칼럼명 (ASC / DESC);
```

☞ 실습 4-4

```
> SELECT * FROM `Sales` ORDER BY `sale`;
> SELECT * FROM `Sales` ORDER BY `sale` ASC;
> SELECT * FROM `Sales` ORDER BY `sale` DESC;
> SELECT * FROM `Member` ORDER BY `name`;
> SELECT * FROM `Member` ORDER BY `name` DESC;
> SELECT * FROM `Member` ORDER BY `rdate` ASC;
> SELECT * FROM `Sales` WHERE `sale` > 50000 ORDER BY `sale` DESC;
> SELECT * FROM `Sales`
  WHERE `sale` > 50000
  ORDER BY `year`, `month`, `sale` DESC;
```

레코드 수를 제한해서 조회

```
SELECT 칼럼명1, 칼럼명2 ... FROM 테이블명 LIMIT 표시할_레코드수;
```

☞ 실습 4-5

```
> SELECT * FROM Sales LIMIT 3;
> SELECT * FROM Sales LIMIT 0, 3;
> SELECT * FROM Sales LIMIT 1, 2;
> SELECT * FROM Sales LIMIT 5, 3;
> SELECT * FROM Sales ORDER BY `sale` DESC LIMIT 3, 5;
> SELECT * FROM Sales WHERE `sale` < 50000 ORDER BY `sale` DESC LIMIT 3;
> SELECT * FROM Sales
  WHERE `sale` > 50000
  ORDER BY `year` DESC, `month`, `sale` DESC
  LIMIT 5;
```

3) MySQL 내장함수

① MySQL 내장함수는 상수나 속성 이름을 입력 값으로 받아 단일 값을 결과로 반환

② 아래 표는 주요 내장함수로 더 많은 함수는 인터넷을 참고

구분	내장함수	설명
집계함수	SUM(칼럼명)	지정한 칼럼의 합계을 구하는 함수
	AVG(칼럼명)	지정한 칼럼의 평균을 구하는 함수
	MAX(칼럼명)	지정한 칼럼의 최대값을 구하는 함수
	MIN(칼럼명)	지정한 칼럼의 최소값을 구하는 함수
	COUNT(칼럼명)	레코드 수를 구한다.
문자열함수	RIGHT('문자열', 2)	'문자열'에서 오른쪽 2문자만 표시
	LEFT('문자열', 2)	'문자열'에서 왼쪽 2문자만 표시
	SUBSTRING('문자열', 2, 3)	'문자열'에서 2번째 문자부터 3개의 문자 표시
날짜함수	NOW()	DATETIME형 칼럼에 현재 날짜와 시간을 입력

칼럼명을 별명으로 사용하기

```
SELECT 칼럼명 AS 별명 FROM 테이블명;
```

☞ 실습 4-6

```
> SELECT SUM(sale) AS `합계` FROM `Sales`;
> SELECT AVG(sale) AS `평균` FROM `Sales`;
> SELECT MAX(sale) AS `최대값` FROM `Sales`;
> SELECT MIN(sale) AS `최소값` FROM `Sales`;
> SELECT COUNT(sale) AS `갯수` FROM `Sales`;
> SELECT COUNT(*) AS `갯수` FROM `Sales`;

> SELECT SUBSTRING(hp, 10, 4) AS '전화번호 끝자리' FROM `Member`;

> INSERT INTO `Member` VALUES ('b101', '을지문덕', '010-5555-1234', '사장', 107, NOW());
```

☞ 실습 4-7. 2018년 1월 매출의 총합을 구하시오.

☞ 실습 4-8. 2019년 2월에 5만원 이상 매출에 대한 총합과 평균을 구하시오.

☞ 실습 4-9. 2020년 전체 매출 가운데 최저, 최고, 매출을 구하시오.

4) 그룹별로 조회하기

그룹별로 조회

```
SELECT 칼럼명1, 칼럼명2 ... FROM 테이블명 GROUP BY 그룹화_칼럼명;
```

☞ 실습 4-10

```
> SELECT * FROM `Sales` GROUP BY `uid`;
> SELECT * FROM `Sales` GROUP BY `year`;
> SELECT * FROM `Sales` GROUP BY `uid`, `year`;
> SELECT `uid`, COUNT(*) AS `건수` FROM `Sales GROUP BY `uid`;
> SELECT `uid`, SUM(sale) AS `합계` FROM `Sales` GROUP BY `uid`;
> SELECT `uid`, AVG(sale) AS `평균` FROM `Sales` GROUP BY `uid`;

> SELECT `uid`, `year`, SUM(sale) AS `합계`
  FROM `Sales`
  GROUP BY `uid`, `year`;

> SELECT `uid`, `year`, SUM(sale) AS `합계`
  FROM `Sales`
  GROUP BY `uid`, `year`
  ORDER BY `year` ASC, `합계` DESC;

> SELECT `uid`, `year`, SUM(sale) AS `합계`
  FROM `Sales`
  WHERE `sale` >= 50000
  GROUP BY `uid`, `year`
  ORDER BY `합계` DESC;
```

그룹화에 검색조건 설정

```
SELECT 칼럼명1, 칼럼명2 ... FROM 테이블명 GROUP BY 그룹화_칼럼명 HAVING 조건;
```

☞ 실습 4-11

```
> SELECT `uid`, SUM(sale) AS `합계` FROM `Sales`
  GROUP BY `uid`
  HAVING `합계` >= 200000;

> SELECT `uid`, `year`, SUM(sale) AS `합계`
  FROM `Sales`
  WHERE `sale` >= 100000
  GROUP BY `uid`, `year`
  HAVING `합계` >= 200000
  ORDER BY `합계` DESC;
```

5) 테이블 합치기

2개 이상의 테이블에서 레코드 조회하기(중복 데이터 제외)

```
SELECT 칼럼명 FROM 테이블명1 UNION SELECT 칼럼명 FROM 테이블명2;
```

☞ 실습 4-12

```
> CREATE TABLE `Sales2` LIKE `Sales`;
> INSERT INTO `Sales2` SELECT * FROM `Sales`;
> UPDATE `Sales2` SET `year` = `year` + 3;

> SELECT * FROM `Sales` UNION SELECT * FROM `Sales2`;
> (SELECT * FROM `Sales`) UNION (SELECT * FROM `Sales2`);
> SELECT `uid`, `year`, `sale` FROM Sales
  UNION
  SELECT `uid`, `year`, `sale` FROM Sales2;

> SELECT `uid`, `year`, SUM(sale) AS `합계`
  FROM `Sales`
  GROUP BY `uid`, `year`
  UNION
  SELECT `uid`, `year`, SUM(sale) AS `합계`
  FROM `Sales2`
  GROUP BY `uid`, `year`
  ORDER BY `year` ASC, `합계` DESC;
```

6) 테이블 결합

① 테이블 결합은 2개 이상의 테이블을 특정 키로 연결해서 데이터를 조회하는 작업을 말한다.
② 테이블 결합에는 내부조인, 외부조인, 셀프조인 등이 있다.
③ 일반적인 테이블결합(JOIN)은 내부조인을 말한다.

내부조인(INNER JOIN)

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

※ INNER 생략 가능

☞ 실습 4-13

```
> SELECT * FROM `Sales` INNER JOIN `Member` ON `Sales`.uid = `Member`.uid;
> SELECT * FROM `Member` INNER JOIN `Department` ON `Member`.dep = `Department`.depNo;

> SELECT * FROM `Sales` AS a JOIN `Member` AS b  ON  a.uid = b.uid;
> SELECT * FROM `Member` AS a JOIN `Department` AS b  ON  a.dep = b.depNo;

> SELECT * FROM `Sales` AS a, `Member` AS b WHERE a.uid = b.uid;
> SELECT * FROM `Member` AS a, `Department` AS b WHERE a.dep = b.depNo;

> SELECT a.`seq`, a.`uid`, `sale`, `name`, `pos` FROM `Sales` AS a
  JOIN `Member` AS b  ON  a.`uid` = b.`uid`;
> SELECT a.`seq`, a.`uid`, `sale`, `name`, `pos` FROM `Sales` AS a
  JOIN `Member` AS b  USING (uid);

> SELECT a.`seq`, a.`uid`, `sale`, `name`, `pos` FROM `Sales` AS a
  JOIN `Member` AS b  ON  a.`uid` = b.`uid`
  WHERE `sale` >= 100000;
> SELECT a.`seq`, a.`uid`, b.`name`, b.`pos`, `year`, SUM(`sale`) AS `합계` FROM `Sales` AS a
  JOIN `Member` AS b  ON a.uid = b.uid
  GROUP BY a.`uid`, a.`year`  HAVING `합계` >= 100000
  ORDER BY a.`year` ASC, `합계` DESC;

> SELECT * FROM `Sales` AS a
  JOIN `Member` AS b  ON a.uid = b.uid
  JOIN `Department` AS c  ON b.dep = c.depNo;
> SELECT a.`seq`, a.`uid`, a.`sale`, b.`name`, b.`pos`, c.`name` FROM `Sales` AS a
  JOIN `Member` AS b  ON a.uid = b.uid
  JOIN `Department` AS c ON b.dep = c.depNo;
> SELECT a.`seq`, a.`uid`, a.`sale`, b.`name`, b.`pos`, c.`name` FROM `Sales` AS a
  JOIN `Member` AS b  ON a.uid = b.uid
  JOIN `Department` AS c  ON b.dep = c.depNo
  WHERE `sale` > 100000
  ORDER BY `sale` DESC;
```

외부조인(LEFT, RIGHT JOIN)

```
SELECT 칼럼 FROM 테이블1 (LEFT | RIGHT) JOIN 테이블2 ON 테이블1.칼럼=테이블2.칼럼;
```

※ 외부조인은 결합하는 테이블의 한쪽(LEFT, RIGHT)의 모든 레코드를 표시한다.

☞ 실습 4-14

```
> SELECT * FROM `Sales` AS a LEFT JOIN `Member` AS b ON a.uid = b.uid;
> SELECT * FROM `Sales` AS a RIGHT JOIN `Member` AS b ON a.uid = b.uid;
> SELECT a.seq, a.uid, `sale`, `name`, `pos`  FROM Sales AS a
  LEFT JOIN Member AS b USING(uid);

> SELECT a.seq, a.uid, `sale`, `name`, `pos`  FROM Sales AS a
  RIGHT JOIN Member AS b USING(uid);
```

☞ 실습 4-15. 모든 직원의 아이디, 이름, 직급, 부서명을 조회하시오.

☞ 실습 4-16. '김유신' 직원의 2019년도 매출의 합을 조회하시오.

☞ 실습 4-17. 2019년 50,000이상 매출에 대해 직원별 매출의 합이 100,000원 이상인 직원이름, 부서명, 직급, 년도, 매출 합을 조회하시오. 단, 매출 합이 큰 순서부터 정렬