



Select

데이터 조회

❖ 기본 형식

SELECT 필드목록

FROM 테이블

[WHERE 조건]

[ORDER BY 정렬기준]

- ✓ 옵션을 생략하면 SELECT 필드 FROM 테이블이 되며 FROM 절의 테이블에서 필드를 읽어 출력
- ✓ SELECT와 FROM 사이의 필드 목록에 출력할 필드의 이름을 지정하되 * 기호를 쓰면 테이블의 모든 필드를 출력









SELECT

1. 모든 열 선택

SELECT 키워드에 “*” 을 사용하여 테이블의 열 데이터 모두를 조회할 수 있습니다.

2. SCOTT이 소유하고 있는 EMP Table의 모든 데이터를 출력

SELECT * FROM emp;

		EMPNO		ENAME		JOB		MGR		HIREDATE		SAL		COMM		DEPTNO
1		7369		SMITH		CLERK		7902		80/12/17		800		(null)		20
2		7499		ALLEN		SALESMAN		7698		81/02/20		1600		300		30
3		7521		WARD		SALESMAN		7698		81/02/22		1250		500		30
4		7566		JONES		MANAGER		7839		81/04/02		2975		(null)		20
5		7654		MARTIN		SALESMAN		7698		81/09/28		1250		1400		30
6		7698		BLAKE		MANAGER		7839		81/05/01		2850		(null)		30
7		7782		CLARK		MANAGER		7839		81/06/09		2450		(null)		10
8		7788		SCOTT		ANALYST		7566		87/04/19		3000		(null)		20
9		7839		KING		PRESIDENT		(null)		81/11/17		5000		(null)		10
10		7844		TURNER		SALESMAN		7698		81/09/08		1500		0		30
11		7876		ADAMS		CLERK		7788		87/05/23		1100		(null)		20
12		7900		JAMES		CLERK		7698		81/12/03		950		(null)		30
13		7902		FORD		ANALYST		7566		81/12/03		3000		(null)		20
14		7934		MILLER		CLERK		7782		82/01/23		1300		(null)		10

데이터 조회

❖ 별명

- ✓ SELECT 명령이 출력하는 내용을 Result Set 또는 Row Set이라고 하는데 형태가 테이블과 동일
- ✓ 원본 테이블의 일부만 읽어도 가로, 세로로 칸이 쳐진 도표 형태가 되기 때문에 SELECT가 만들어 내는 Result Set을 하나의 테이블처럼 사용
- ✓ Result Set의 필드 캡션은 테이블에서 정의한 이름과 동일한데 name 필드는 name이라고 출력하고 popu는 popu라고 출력하는데 테이블의 필드명은 구분 가능하고 입력하기 쉬운 짧은 명칭일 뿐이어서 사용자가 읽기에는 직관적이지 못한 경우가 있을 수 있는데 이럴 때는 필드에 대한 별명을 지정하며 Result Set의 헤더에 필드 이름 대신 별명을 출력
- ✓ 별명은 어디까지나 문자열일 뿐이므로 명칭 규칙에 영향을 받지 않는데 공백이나 기호는 물론이고 모든 문자를 자유롭게 표기할 수 있음
- ✓ 필드에 별명을 붙일 때는 다음 형식을 사용
필드명 [AS] "별명"
- ✓ 필드명과 별명 사이에 전치사 AS를 넣는데 생략해도 상관없음
- ✓ 별명은 명칭이 아니며 공백이나 특수문자를 포함할 수 있어 큰 따옴표로 감싸져 평이한 단어라면 따옴표를 생략해도 상관없음

데이터 조회

❖ 별명

✓ 별명 사용

```
SELECT ename AS 이름, sal AS "급여(달러)", hiredate '입사일' FROM  
emp;
```

이름	급여(달러)	입사일
SMITH	800.00	1980-12-17
ALLEN	1600.00	1981-02-20
WARD	1250.00	1981-02-22
JONES	2975.00	1981-04-02
MARTIN	1250.00	1981-09-28
BLAKE	2850.00	1981-05-01

데이터 조회

❖ 계산 값의 출력

- ✓ 필드 목록에 계산식을 사용하면 테이블에 저장된 값을 가공하여 출력
- ✓ sal 필드에 10000을 곱해서 출력

```
SELECT ename AS 이름, sal * 1000 AS "급여" FROM emp;
```

이름	급여
'SMITH'	'800000.00'
'ALLEN'	'1600000.00'
'WARD'	'1250000.00'
'JONES'	'2975000.00'
'MARTIN'	'1250000.00'
'BLAKE'	'2850000.00'

데이터 조회

❖ 계산 값의 출력

✓ 계산식 출력

```
SELECT 60 * 60 * 24;
```

86,400

✓ 연습문제

- SELECT 문을 사용하여 1년은 몇 초인지 계산

WHERE 조건과 비교 연산자

원하는 로우만 얻으려면 다음과 같이 로우를 제한하는 조건을 SELECT 문에 WHERE 절을 추가하여 제시해야 합니다.

형식	SELECT * [column1, column2, .. ,columnn] FROM table_name WHERE 조건절;
----	--

조건 절은 다음의 세 부분으로 구성이 됩니다.

조건절의 구성	WHERE SAL >= 3000; ① 컬럼 ② 연산자 ③ 비교대상값
---------	--

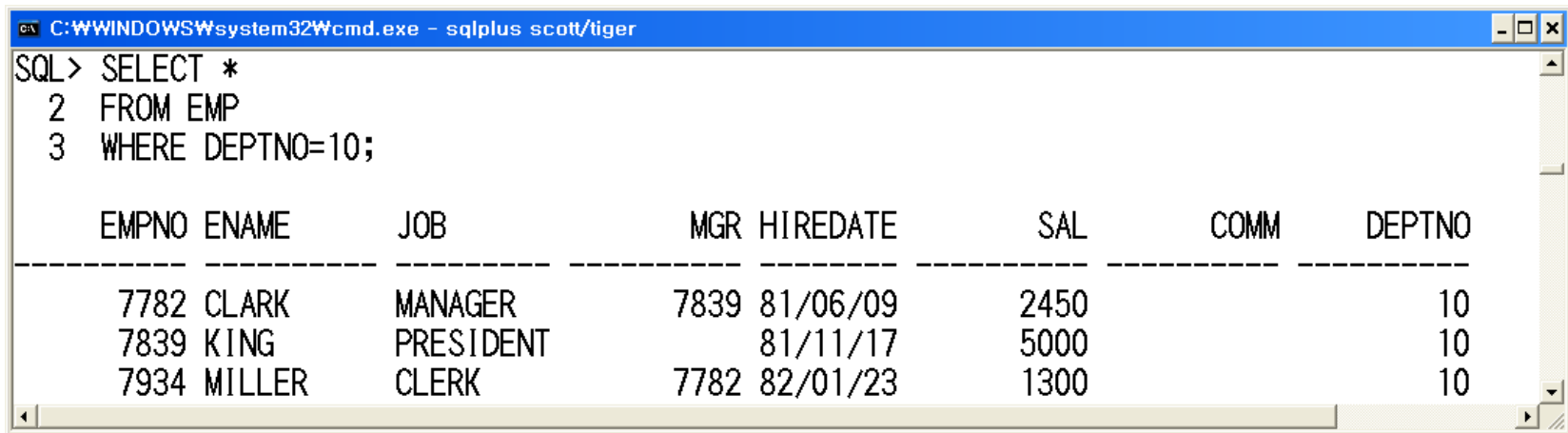
비교 연산자

연산자	의 미	예 제
=	같다.	SELECT EMPNO, ENAME, SAL FROM EMP WHERE SAL=3000;
>	보다 크다.	SELECT EMPNO, ENAME, SAL FROM EMP WHERE SAL>3000;
<	보다 작다.	SELECT EMPNO, ENAME, SAL FROM EMP WHERE SAL<3000;
>=	보다 크거나 같다.	SELECT EMPNO, ENAME, SAL FROM EMP WHERE SAL>=3000;
<=	보다 작거나 같다.	SELECT EMPNO, ENAME, SAL FROM EMP WHERE SAL<=3000;
<>, !=, ^=	다르다.	SELECT EMPNO, ENAME, SAL FROM EMP WHERE SAL<>3000;

비교 연산자

예

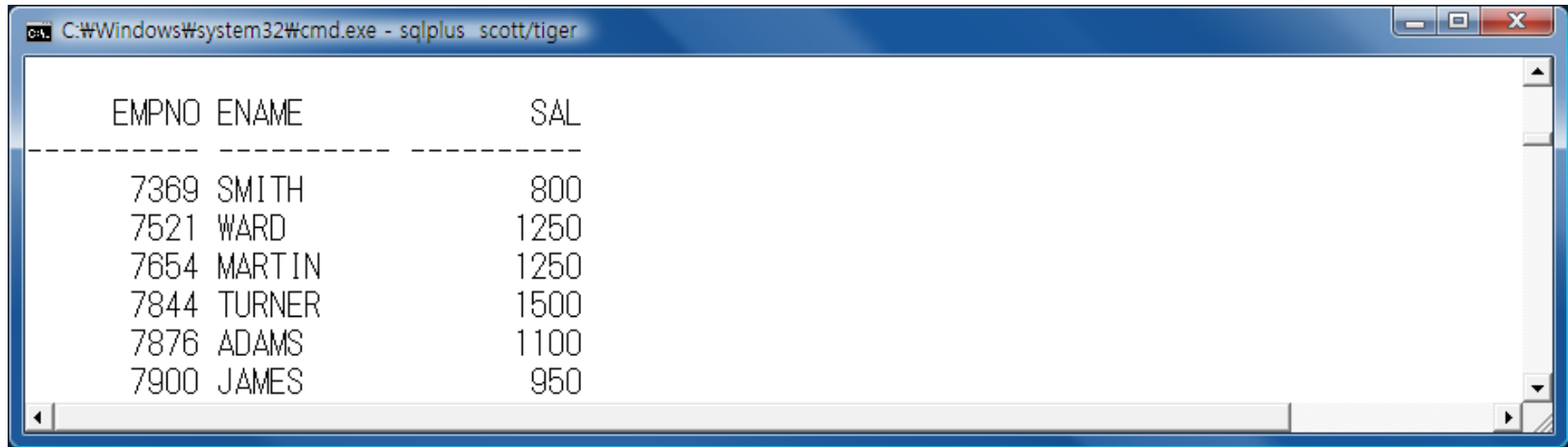
```
SELECT *  
FROM EMP  
WHERE DEPTNO=10;
```



EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7782	CLARK	MANAGER	7839	81/06/09	2450		10
7839	KING	PRESIDENT		81/11/17	5000		10
7934	MILLER	CLERK	7782	82/01/23	1300		10

비교 연산자

1.급여가 1500 이하인 사원의 사원번호, 사원 이름, 급여를 출력하는 SQL 문을 작성해 보시오.



EMPNO	ENAME	SAL
7369	SMITH	800
7521	WARD	1250
7654	MARTIN	1250
7844	TURNER	1500
7876	ADAMS	1100
7900	JAMES	950

<힌트> 사원 정보가 저장된 테이블의 이름은 EMP이고, 사원번호 컬럼은 EMPNO, 사원이름 컬럼은 ENAME, 급여 컬럼은 SAL입니다.

데이터 조회

❖ 조건문

- ✓ 다르다를 의미하는 <> 와 같지 않다를 의미하는 !=는 같은 연산자
- ✓ 숫자는 상수를 그냥 쓰지만 문자열과 날짜 상수는 항상 작은 따옴표로 감싸야 함
- ✓ 도시명이 서울이라는 조건은 아래처럼 작성하는데 MySQL 과 MariaDB는 큰 따옴표로 감싸도 에러는 아님

`SELECT * FROM emp WHERE name = 'SCOTT';`

- ✓ SQL 문 자체는 대소문자를 가리지 않는데 Mari 키워드나 테이블명, 필드명은 대소문자 구분없이 작성해도 되지만 필드 안에 저장된 값은 대소문자를 구분
- ✓ Maria DB 와 MySQL 은 저장할 때는 대소문자를 구분하지만 비교할 때는 대소문자 구별을 하지 않음

데이터 조회

❖ 조건문

✓ NULL 비교

- NULL은 값이 입력되어 있지 않은 특수한 상태를 표현
- 값을 알 수 없거나 아직 결정할 수 없다는 의미이며 0 과는 다름
- NULL 여부를 표현할 때는 = 나 <>, != 대신에 is null 과 is not null 을 사용
- 테이블에서 NULL 여부 표현 - 첫번째는 데이터가 조회되지 않음

SELECT * FROM emp WHERE comm = NULL;

SELECT * FROM emp WHERE comm IS NULL;

SELECT * FROM emp WHERE com IS NOT NULL;

SELECT empno,ename,job,sal,comm,deptno FROM emp WHERE
comm IS NULL

Results:

	EMPNO	ENAME	JOB	SAL	COMM	DEPTNO
1	7369	SMITH	CLERK	800	(null)	20
2	7566	JONES	MANAGER	2975	(null)	20
3	7698	BLAKE	MANAGER	2850	(null)	30
4	7782	CLARK	MANAGER	2450	(null)	10
5	7788	SCOTT	ANALYST	3000	(null)	20
6	7839	KING	PRESIDENT	5000	(null)	10
7	7876	ADAMS	CLERK	1100	(null)	20
8	7900	JAMES	CLERK	950	(null)	30
9	7902	FORD	ANALYST	3000	(null)	20
10	7934	MILLER	CLERK	1300	(null)	10

데이터 조회

❖ 조건문

✓ 논리 연산자

- 두 개 이상의 조건을 동시에 점검할 때는 AND, OR 논리 연산자를 사용
- AND는 두 조건이 모두 참 인 레코드를 검색하며 OR는 두 조건 중 하나라도 참 인 레코드를 검사
- AND 의 우선 순위가 OR 보다 높음
- NOT 연산자는 표현식의 진위 여부를 반대로 변경
- 인구 100만 이상 그리고 면적 700 이상인 도시를 조회

```
SELECT * FROM tCity WHERE popu >= 100 AND area >= 700;
```

- 3개 이상의 조건도 동일

```
SELECT * FROM tCity WHERE region = '경기'  
AND popu >= 50 OR area >= 500;
```

```
SELECT * FROM tCity  
WHERE region = '경기' AND (popu >= 50 OR area >= 500);
```

데이터 조회

❖ 조건문

✓ 논리 연산자

● NOT

```
SELECT * FROM emp WHERE job != 'MANAGER';
```

```
SELECT * FROM emp WHERE NOT(job = 'MANAGER');
```

```
SELECT * FROM emp WHERE job != 'MANAGER' AND  
deptno != 10;
```

```
SELECT * FROM emp WHERE NOT(job = 'MANAGER' OR  
deptno = 10);
```

✓ 연습문제

- 직원 목록에서 월급이 3000 미만이면서 부서코드 10 인 직원이 누구인지 조회
- 20번 부서의 MANAGER 이름을 조회

데이터 조회

❖ 조건문

✓ LIKE

- = 비교 연산자는 완전히 일치하는 조건식을 표현하는데 비해 LIKE 연산자는 패턴으로 부분 문자열을 검색
- LIKE 문 의 패턴에는 와일드 카드를 사용
- 와일드 카드

문자	설명
%	임의 개수의 임의 문자
_	하나의 임의 문자

- 와일드 카드 문자를 검색할 검색하고자 할 때는 ESCAPE 를 이용

데이터 조회

❖ 조건문

✓ LIKE

```
SELECT empno,ename,job,sal,hiredate,deptno FROM emp  
where hiredate LIKE '82%';
```

```
SELECT empno, ename, mgr, sal  
FROM emp  
WHERE ename Like 'W%';
```

```
SELECT empno, ename, mgr, sal  
FROM emp  
WHERE ename Like '%N';
```

```
SELECT empno, ename, mgr, sal  
FROM emp  
WHERE ename Like '_A%';
```

데이터 조회

❖ 조건문

✓ LIKE

- sale에 30% 가 포함된 데이터 조회

WHERE sale LIKE '%30#%' ESCAPE '#';

이것은 임의 문자와
대응되는 와일드 카드

이것은 그냥
백분율 기호

이스케이프 문자

LIKE '%30# %' ESCAPE '#'

✓ 연습문제

- 직원 목록에서 'T' 가 포함된 이름 조회
- 이름에 'S' 자가 포함된 직원을 조회

데이터 조회

❖ 조건문

✓ BETWEEN

- BETWEEN ~ AND 문은 BETWEEN 최소값 AND 최대값 형식으로 두 값 사이의 범위를 제한
- 범위 조건은 수치값에 대해 사용하지만 문자열이나 날짜 등에도 사용할 수 있음
- 급여가 1000 ~ 3000 사이인 이름을 조회

```
SELECT *  
FROM emp  
WHERE sal BETWEEN 1000 AND 3000;
```

```
SELECT *  
FROM emp  
WHERE sal >= 1000 AND sal <= 3000;
```

데이터 조회

❖ 조건문

✓ BETWEEN

- 직원의 이름과 입사일로 범위 검색을 수행

```
SELECT * FROM emp
```

```
WHERE name BETWEEN 'E' AND 'T';
```

```
SELECT * FROM emp
```

```
WHERE hiredate BETWEEN '19800101' AND '19811231';
```

✓ 연습문제

- 급여가 1500~2500 사이의 직원 목록을 조회
- 부서코드가 20인 직원의 목록을 조회

데이터 조회

❖ 조건문

✓ IN

- IN 연산자는 불연속적인 값 여러 개의 목록을 제공하여 이 목록과 일치하는 레코드를 검색
- IN 연산자 뒤의 괄호 안에 콤마로 구분된 값 목록을 나열하여 이 중 하나에 해당하는지 점검하는데 값 개수에는 제한이 없어 얼마든지 많은 값을 넣을 수 있음
- IN 연산자는 불연속적인 값 여러 개의 목록을 제공하여 이 목록과 일치하는 레코드를 검색
- IN 연산자 뒤의 괄호 안에 콤마로 구분된 값 목록을 나열하여 이 중 하나에 해당하는지 점검하는데 값 개수에는 제한이 없어 얼마든지 많은 값을 넣을 수 있음
- NOT 과 결합해서 사용 가능
- job 필드가 SALESMAN 또는 MANAGER 인 모든 직원을 조회

```
SELECT *
```

```
FROM emp
```

```
WHERE job IN ('MANAGER', 'SALESMAN');
```

데이터 조회

❖ 조건문

✓ IN

- job 필드가 SALESMAN 또는 MANAGER가 아닌 모든 직원을 조회

```
SELECT * FROM emp
```

```
WHERE job NOT IN ('MANAGER', 'SALESMAN');
```

- ename 이 A이 와 S로 시작하는 단어를 조회

```
SELECT * FROM emp
```

```
WHERE ename LIKE 'A%' OR ename LIKE 'S%';
```

✓ 연습문제

- 10부서나 30번에 근무하는 직원의 목록 조회
- 급여가 3000이거나 5000인 직원 목록 조회

데이터 조회

❖ 정렬

✓ ORDER BY

- SELECT 명령에 별 지정이 없을 경우 레코드의 출력 순서는 DBMS의 디폴트 순서를 따름
- 오라클은 레코드의 입력 순서를 기억해 두고 그대로 가져오고 SQL Server 와 MariaDB는 기본키에 대해 오름차순으로 정렬
- 관계형 DB에서 레코드의 물리적인 순서는 큰 의미가 없는 대신 출력할 때 ORDER BY 절로 정렬 순서를 원하는 대로 지정
- 기본 형식
ORDER BY 필드 [ASC | DESC]
 - ◆ 오름차순의 경우는 ASC 그리고 내림차순의 경우는 DESC를 지정
 - ◆ 기본값이 ASC 이므로 ASC는 생략 가능
- 2개 이상의 기준 필드를 지정할 수 있는데 이 경우에는 첫번째 기준 필드의 값이 같으면 두번째 기준 필드를 비교하여 정렬 순서를 결정
- 필드 이름 대신에 필드의 순서 값을 설정하는 것도 가능
- 정렬 기준 필드를 조회하지 않는 것도 가능 - 권장하지는 않음
- 계산식을 이용한 정렬 가능

데이터 조회

❖ 정렬

✓ ORDER BY

- 급여에 따른 정렬

```
SELECT * FROM emp ORDER BY sal;
```

```
SELECT * FROM tCity ORDER BY popu DESC;
```

- 부서 별로 정렬하고 같은 부서가 같으면 이름의 내림차순으로 조회

```
SELECT * FROM emp ORDER BY deptno, ename DESC;
```

- 필드 순서를 이용한 정렬

```
SELECT empno, ename, sal FROM emp ORDER BY 2;
```

✓ ORDER BY

- WHERE를 포함한 정렬

```
SELECT * FROM emp WHERE deptno = '20' ORDER BY sal;
```

✓ 연습문제

- 직원 목록을 월급이 적은 사람부터 순서대로 출력하되 월급이 같다면 입사일 순서로 조회
- 30번 부서 직원을 먼저 입사한 순서대로 정렬해서 조회

데이터 조회

❖ DISTINCT

- ✓ 중복을 제거할 때 사용
- ✓ SELECT 절의 맨 앞에 기재하며 뒤에 여러 개의 필드를 나열 할 수 있는데 여러 개의 필드를 나열하면 모든 필드의 값이 일치하는 경우 제외
- ✓ 테이블에서 deptno 의 값을 중복을 제외하고 조회
SELECT deptno FROM emp;
- ✓ 연습문제
 - 1981년 이후에 신입 사원을 받은 적이 있는 부서 목록을 조회

데이터 조회

❖ 행의 개수 제한

✓ LIMIT

- SELECT 구문의 맨 뒤에 기재해서 행의 개수를 제한
- ORACLE에서는 ROWNUM 이라는 Pseudo Column 을 이용하고 MS-SQL에서는 TOP을 이용
- 기본 형식
LIMIT [건너뛴 개수], 조회할 개수
 - ◆ 건너뛴 개수를 생략하면 0
 - ◆ LIMIT 조회할 개수 OFFSET 건너뛴 개수 형태로 입력하는 것도 가능
- 급여가 큰 상위 4명의 이름과 급여 조회
SELECT * FROM emp ORDER BY sal DESC LIMIT 4;
- 급여가 큰 앞의 2개는 건너뛰고 이후 3개의 데이터 조회
SELECT * FROM emp ORDER BY sal DESC LIMIT 2, 3;
SELECT * FROM emp ORDER BY sal DESC LIMIT 3 OFFSET 2 ;

데이터 조회

❖ 행의 개수 제한

✓ OFFSET FETCH

- ORDER BY 의 옵션으로 설정하는 표준적인 행의 개수 제한 방법

- 기본 형식

ORDER BY 기준필드 OFFSET 건너뛴 행 수 ROWS FETCH NEXT 출력할 행 수 ROWS ONLY

◆ ROWS는 ROW라고 써도 되며 NEXT는 FIRST라고 써도 됨

- 급여 순으로 상위 4개의 이름과 급여 정보 조회

```
SELECT * FROM emp ORDER BY sal DESC OFFSET 0 ROWS  
FETCH NEXT 4 ROWS ONLY;
```

- 급여 순으로 상위 2개를 건너뛰고 3개의 보 조회

```
SELECT * FROM emp ORDER BY sal DESC OFFSET 2 ROWS  
FETCH NEXT 3 ROWS ONLY;
```

✓ 연습문제

- 직원을 월급순으로 내림차순 정렬한 후 12위에서 16위까지 조회

데이터 집계

❖ 집계 함수

- ✓ 데이터를 그룹화 해서 통계를 계산해주는 함수
- ✓ GROUP BY 이후에 그룹화가 이루어지므로 HAVING, SELECT 등에서만 사용 가능
- ✓ GROUP BY 절의 그룹화 항목 하고만 같이 조회 가능
- ✓ COUNT
 - 개수를 세는 함수
 - 필드명이나 *를 지정
 - emp 테이블의 데이터 개수 조회

```
SELECT COUNT(*) FROM emp;
```

```
SELECT COUNT(*) AS "총 직원수" FROM emp;
```
 - 조건과 함께 사용

```
SELECT COUNT(*) FROM tStaff WHERE sal >= 2000;
```

```
SELECT COUNT(*) FROM tStaff WHERE sal >= 1500;
```

데이터 집계

❖ 집계 함수

✓ COUNT

● 필드 이름 사용

```
SELECT COUNT(ename) FROM emp;
```

```
SELECT COUNT(deptno) FROM emp;
```

```
SELECT COUNT(DISTINCT deptno) FROM emp;
```

```
SELECT COUNT(sal) FROM emp;
```

```
SELECT COUNT(*) - COUNT(comm) FROM emp;
```

```
SELECT COUNT(*) FROM emp WHERE comm IS NULL;
```

✓ 연습문제

- Comm이 null 없는 직원은 목록을 조회

- 급여가 2000 이상인 직원이 몇 명이나 되는지 조회

데이터 집계

❖ 일반 집계 함수

✓ 집계 함수

- SUM: 합계
- AVG: 평균
- MIN: 최소값
- MAX: 최대값
- STDDEV: 표준 편차
- VARIANCE: 분산

✓ 문자열의 합계 나 평균은 구할 수 없지만 최소값 과 최대값을 조회할 수 있음

✓ 급여의 총합 과 평균을 조회

```
SELECT SUM(sal), AVG(sal) FROM emp;
```

✓ 급여가 최소값 과 최대값 조회

```
SELECT MIN(sal), MAX(sal) FROM emp;
```

✓ WHERE 절 사용

```
SELECT SUM(sal), AVG(sal) FROM tStaff WHERE job = 'SALESMAN';
```

```
SELECT MIN(sal), MAX(sal) FROM tStaff WHERE deptno = 20;
```

데이터 집계

❖ 일반 집계 함수

✓ 문자열 사용

```
SELECT MIN(ename) FROM emp;
```

```
SELECT MAX(ename), name FROM emp;
```

❖ 집계 함수 와 NULL

✓ NULL은 값을 알 수 없는 특수한 상태이므로 모든 집계 함수는 NULL을 무시하고 통계를 계산

✓ COUNT는 일치하는 데이터가 없으면 0을 리턴하지만 다른 집계함수는 일치하는 데이터가 없으면 NULL을 리턴

✓ salary 의 평균 조회

```
SELECT AVG(sal) FROM emp;
```

```
SELECT SUM(sal)/COUNT(*) FROM emp;
```

✓ NULL 조회

```
SELECT AVG(sal) FROM emp;
```

```
SELECT SUM(sal)/COUNT(*) FROM emp;
```

```
SELECT COUNT(*) FROM emp WHERE depno = 20;
```

```
SELECT MAX(sal) FROM emp WHERE deptno = 10;
```

데이터 집계

❖ GROUPING

✓ GROUP BY

- 기준이 되는 필드를 뒤에 적어 주면 기준 필드가 같은 레코드를 모아 통계 값을 구함
- 기준 필드는 집계 함수와 같이 쓸 수 있어 목록도 보기 좋게 출력할 수 있음
- GROUP BY의 기준 필드는 중복값이 있을 때만 의미가 있는데 레코드 별로 고유한 값을 가지는 필드는 그룹핑 기준으로 부적합하며 구분이나 분류 필드가 적합
- 2개 이상의 필드를 기준으로 그룹핑 가능
- 부서별 평균 월급을 구하려면 depart 필드 기준으로 그룹핑

```
SELECT deptno, AVG(sal) FROM emp GROUP BY deptno;
```
- 여러 개의 집계 함수 사용

```
SELECT deptno, COUNT(*), MAX(sal), AVG(sal)  
FROM emp GROUP BY deptno;
```

✓ 연습문제

- Emp table에서 부서별 인원수를 조회

데이터 집계

❖ GROUPING

✓ HAVING

- GROUP BY 다음에 오며 통계 결과 중 출력할 그룹의 조건을 지정

```
SQL> SELECT      deptno, max(sal)
2  FROM          emp
3  GROUP BY      deptno
4  HAVING        max(sal)>2900;
```

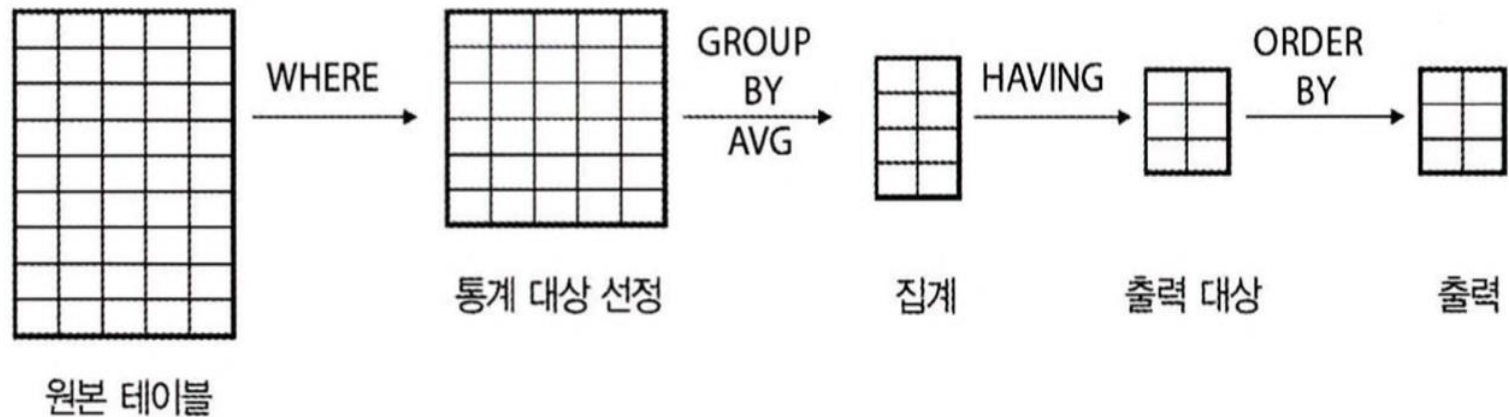
DEPTNO	MAX (SAL)
10	5000
20	3000

데이터 집계

❖ GROUPING

✓ HAVING

- 쿼리가 실행되는 과정



✓ 연습문제

- 각 부서에서 가장 급여가 많은 이름과 급여를 조회하는데 급여가 2000만 이상인 사람만 대상
- 각 부서 평균 급여를 구하되 평균 급여가 2000 이상인 부서만 조회

논리 함수

- ▶ IF(논리식, 참일 때 값, 거짓일 때 값)

Select empno ,ename,sal,if(sal>=1500,'good','poor' as result from emp;

- ▶ IFNULL(값1,값2)

값1이 NULL 이면 값2로 대체하고 그렇지 않으면 값1을 출력

select empno, ename, sal, sal + ifnull(comm,0) from personal

- ▶ select DATABASE() : 현재의 데이터베이스 이름을 출력한다.

- ▶ mysql5.* : select password('문자열') : 문자열을 암호화한다.

- ▶ mysql8.* : select sha('a');

- ▶ FORMAT(숫자, 소수이하자리수) : 숫자를 #,###,###.## 형식으로 출력

--임의의 소수점자리수를 생성한다./소숫점을 필요한 만큼 취한다.

--소숫점을 만들어 같은 길이로 불러와서 소숫점을 버리고 출력하는 등에 응용

select format(123,5);

select format(123.12345600123,9);

select format(123.123,-3);

※ 소숫점이하 자리수가 0 이나 음수 값은 해당이 안됨