



DDL : Data Definition Language

데이터 정의어(Data Definition Language)

데이터 정의어(이하 DDL)는 데이터 간에 관계를 정의하여 데이터베이스 구조를 설정하는 SQL 문장

구분	명령어	설명
데이터베이스	CREATE DATABASE	데이터베이스를 생성한다.
	ALTER DATABASE	데이터베이스를 변경한다.
테이블	CREATE TABLE	테이블을 생성한다.
	ALTER TABLE	테이블을 변경한다.
	DROP TABLE	테이블을 제거한다.
테이블 스페이스	CREATE TABLESPACE	테이블 스페이스를 생성한다.
	ALTER TABLESPACE	테이블 스페이스를 변경한다.
	DROP TABLESPACE	테이블 스페이스를 제거한다.

데이터 정의어(Data Definition Language)

구분	명령어	설명
인덱스	CREATE INDEX	인덱스를 생성한다.
	ALTER INDEX	인덱스를 변경한다.
	DROP INDEX	인덱스를 제거한다.
뷰	CREATE VIEW	뷰를 생성한다.
	ALTER VIEW	뷰를 변경한다.
	DROP VIEW	뷰를 제거한다.
동의어	CREATE SYNONYM	동의어를 생성한다.
	DROP SYNONYM	동의어를 제거한다.
사용자	CREATE USER	사용자를 생성한다.
	ALTER USER	사용자를 변경한다.
	DROP USER	사용자를 제거한다.

데이터 정의어(Data Definition Language)

구분	명령어	설명
함수	CREATE FUNCTION	함수를 생성한다.
	ALTER FUNCTION	함수를 변경한다.
	DROP FUNCTION	함수를 제거한다.
프러시저	CREATE PROCEDURE	프러시저를 생성한다.
	ALTER PROCEDURE	프러시저를 변경한다.
	DROP PROCEDURE	프러시저를 제거한다.
타입	CREATE TYPE	타입을 생성한다.
	ALTER TYPE	타입을 변경한다.
	DROP TYPE	타입을 제거한다.

데이터 정의어(Data Definition Language)

구분	명령어	설명
권한	GRANT	사용자에게 특권을 부여한다.
	REVOKE	사용자에게 특권을 회수한다.
역할	CREATE ROLE	역할을 생성한다.
	ALTER ROLE	역할을 변경한다.
	DROP ROLE	역할을 제거한다.
객체	RENAME	테이블, 뷰, 동의어, 시퀀스 등의 스키마 객체의 이름을 변경한다.

Create Tables to Store Data

테이블 인스턴스 - 테이블의 구조와 칼럼의 특성을 요약

Symbols	Explanation
PK	기본 키 열
FK	외래 키 열
FK1, FK2	동일한 테이블에 있는 두 개의 외부 키
NN	NOT NULL 열
U	고유 열

Create Tables to Store Data

Table Instance Chart

Table Name: E_EMP

Column Name	ID	LAST_NAME	FIRST_NAME	START_DATE	SALARY	MANAGER_ID	DEPT_ID
Key Type	PK					FK1	FK2
Nulls / Unique	NN, U					NN	
FK Ref Table						s_emp	S_DEPT
FK Ref Column						ID	ID
Data Type	NUMBER	CHAR	CHAR	DATE	NUMBER	NUMBER	NUMBER
Maximum Length	7	25	25		11	7	7
Sample Data	1	Alexander	Lee	2022-09-01	1500		10
	2	Jonathan	Hong	2022-10-05	1000	1	20

Create Tables to Store Data

Syntax

```
CREATE TABLE [user.] table_name
    ({column_name datatype | table_constraint
    ,column_name datatype | table_constraint});
```

where	<i>table_name</i>	테이블 이름
	<i>column_name</i>	열 이름
	<i>datatype</i>	데이터 타입
	<i>table_constraint</i>	제약 조건

Create Tables to Store Data

스키마 객체 이름

- 이름 부여 시 따옴표(“ ”) 없는 식별자는 모두 대문자로 간주
- 따옴표 있는 식별자는 대소문자를 구분

모두 다른 식별자	모두 같은 식별자
department “department” “Department”	department DEPARTMENT “DEPARTMENT”

식별자를 기술 할 때 규칙

- 길이가 30bytes를 넘으면 안된다.
- 따옴표 없는 식별자는 알파벳, 한글, 숫자, 언더바(_), \$, #만 사용. 단, 숫자, '\$', '#'는 첫 글자로 올 수 없다.
- 따옴표 있는 식별자는 공백을 포함한 어떤 문자 사용 가능. 다만, 큰따옴표(“ ”)는 사용 불가능.
- 하나의 네임스페이스 안에 서로 다른 두 객체가 동일한 이름을 사용할 수 없다.

Create Tables to Store Data

MySQL에서 제공하는 데이터 타입

구분	데이터 타입
문자형	CHAR, VARCHAR, TEXT, MEDIUMTEXT, LONGTEXT, JASON
숫자형	TINYINT, SMALLINT, MEDIUMINT, INT, BIGINT, FLOAT, DECIMAL, DOUBLE
날짜형	DATE, TIME, DATETIME, TIMESTAMP, YEAR
이진 데이터타입	BINARY, VARBINARY, TINYBLOB, BLOB,

Create Tables to Store Data

데이터 무결성 제약 조건

Constraint	Description
NOT NULL	NULL값을 허용하지 않고, 반드시 데이터를 입력. 제약조건이 NULL이면 해당 컬럼은 NULL값을 허용.
UNIQUE	해당 칼럼이 중복되는 데이터가 존재할 수 없는 유일성을 보장하는 제약조건
PRIMARY KEY	NOT NULL, UNIQUE 제약 조건의 결합과 같다. 테이블 또는 뷰는 단 한 개의 PRIMARY KEY 제약조건을 가질 수 있다.
FOREIGN KEY	같은 테이블 또는 서로 다른 두 개 테이블의 키 컬럼 사이의 관계

Create Tables to Store Data

EXAMPLE

Table Name: S_REGION

Column Name	ID	NAME
Key Type	PK	
Nulls/Unique	NN, U	NN, U
Sample data	1	North America
	2	South America
	3	Africa/Middle East
	4	Asia
	5	Europe

```
SQL> CREATE TABLE s_region
2   ( id                INT PRIMARY KEY,
3   name                VARCHAR (50) NOT NULL UNIQUE);
Table 'S_REGION' created.
```

Create Tables to Store Data

EXAMPLE

Table Name: S_DEPT

Column Name	deptno	dname	loc
Key Type	PK		
Nulls/Unique			
Datatype	INT	VARCHAR	VARCHAR
Maximum Length	2	14	13
Sample data	10	ACCOUNTING	NEW YORK
	20	RESEARCH	DALLAS
	30	SALES	CHICAGO
	40	OPERATIONS	BOSTON

```
SQL> CREATE TABLE s_dept
2  ( deptno          INT PRIMARY KEY,
3   dname            VARCHAR (14),
4   loc              VARCHAR (13));
Table 'S_DEPT' created.
```

Create Tables to Store Data

EXAMPLE

Table Name: S_DEPT

```
INSERT INTO s_dept VALUES (10, 'ACCOUNTING', 'NEW YORK');  
INSERT INTO s_dept VALUES (20, 'RESEARCH', 'DALLAS');  
INSERT INTO s_dept VALUES (30, 'SALES', 'CHICAGO');  
INSERT INTO s_dept VALUES (40, 'OPERATIONS', 'BOSTON');
```

Create Tables to Store Data

EXAMPLE

Table Name: s_emp

Column Name	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
Key Type	PK			FK1				FK2
Nulls/Unique	NN, U	NN						
FK Ref Table				s_emp				S_DEPT
FK Ref Column				EMPNO				DEPTNO
Datatype	INT	VARCHAR	VARCHAR	INT	DATE	INT	INT	INT
MaxLength	4	10	9	4		7	7	2
Sample data	7839	KING	PRESIDENT	null	1981-11-17	5000	null	10
	7566	JONES	MANAGER	7839	1981-02-04	2975	null	20
	7902	FORD	ANALYST	7566	1981-03-12	3000	null	20

Create Tables to Store Data

```
SQL> create table s_emp  
( empno INT PRIMARY KEY,  
  ename VARCHAR (10) NOT NULL,  
  job VARCHAR (9),  
  mgr INT ,  
  hiredate DATE,  
  sal INT ,  
  comm INT ,  
  deptno INT,  
  FOREIGN KEY (mgr) REFERENCES s_emp(empno) on update cascade,  
  FOREIGN KEY (deptno) REFERENCES s_dept(deptno));
```


Create Tables to Store Data

EXAMPLE

Table Name: s_emp

```
INSERT INTO s_emp VALUES (7839,'KING','PRESIDENT',NULL,'81-11-17',5000,NULL,10);
INSERT INTO s_emp VALUES (7698,'BLAKE','MANAGER',7839,'81-05-01',2850,NULL,30);
INSERT INTO s_emp VALUES (7782,'CLARK','MANAGER',7839,'81-05-09',2450,NULL,10);
INSERT INTO s_emp VALUES (7566,'JONES','MANAGER',7839,'81-04-01',2975,NULL,20);
INSERT INTO s_emp VALUES (7654,'MARTIN','SALESMAN',7698,'81-09-10',1250,1400,30);
INSERT INTO s_emp VALUES (7499,'ALLEN','SALESMAN',7698,'81-02-11',1600,300,30);
INSERT INTO s_emp VALUES (7844,'TURNER','SALESMAN',7698,'81-08-21',1500,0,30);
INSERT INTO s_emp VALUES (7900,'JAMES','CLERK',7698,'81-12-11',950,NULL,30);
INSERT INTO s_emp VALUES (7521,'WARD','SALESMAN',7698,'81-02-23',1250,500,30);
INSERT INTO s_emp VALUES (7902,'FORD','ANALYST',7566,'81-12-11',3000,NULL,20);
INSERT INTO s_emp VALUES (7369,'SMITH','CLERK',7902,'80-12-09',800,NULL,20);
INSERT INTO s_emp VALUES (7788,'SCOTT','ANALYST',7566,'82-12-22',3000,NULL,20);
INSERT INTO s_emp VALUES (7876,'ADAMS','CLERK',7788,'83-01-15',1100,NULL,20);
INSERT INTO s_emp VALUES (7934,'MILLER','CLERK',7782,'82-01-11',1300,NULL,10);
```

CONFIRM THE STRUCTURE OF A TABLE

테이블 구조 확인

Syntax

```
DESCRIBE table_name
```

Example

```
mysql> DESCRIBE s_emp;
```

```
+-----+-----+-----+-----+-----+-----+
| Field | Type   | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| empno | int    | NO   | PRI | NULL    |       |
| ename | varchar(10) | NO   |     | NULL    |       |
| job   | varchar(9) | YES  |     | NULL    |       |
| mgr   | int    | YES  | MUL | NULL    |       |
| hiredate | date  | YES  |     | NULL    |       |
| sal   | int    | YES  |     | NULL    |       |
| comm  | int    | YES  |     | NULL    |       |
| deptno | int    | YES  | MUL | NULL    |       |
+-----+-----+-----+-----+-----+-----+
8 rows in set (0.01 sec)
```

ADD A COLUMN TO A TABLE

테이블에 칼럼 추가하기

Syntax

```
ALTER TABLE table_name  
ADD ( column_name datatype  
[, column_name datatype]...)
```

Example

```
SQL> ALTER TABLE s_region  
2 ADD COLUMN (comments VARCHAR (255));  
Table 'S_REGION' altered.
```

MODIFY A COLUMN

테이블에 존재하는 칼럼 속성 변경
데이터 타입, 기본값, 제약조건 변경

Syntax

```
ALTER TABLE table_name  
MODIFY ( column_name datatype  
[, column_name datatype]...)
```

Example

s_emp 테이블의 ENAME 칼럼의 길이 20으로 변경 및 SAL 값은 NULL이 될 수 없음

```
SQL> ALTER TABLE s_emp  
2 MODIFY ename VARCHAR (20);  
Table 's_emp' altered.
```

```
SQL> ALTER TABLE s_emp  
2 MODIFY SAL INT NOT NULL;  
Table 's_emp' altered.
```

ADD AND REMOVE DATA CONSTRAINTS

Example

s_emp 테이블의 제약조건 조회

```
select * from information_schema.table_constraints where table_name='s_emp';
```

```
mysql> select * from information_schema.table_constraints where table_name='s_emp';
```

CONSTRAINT_CATALOG	CONSTRAINT_SCHEMA	CONSTRAINT_NAME	TABLE_SCHEMA	TABLE_NAME	CONSTRAINT_TYPE	ENFORCED
def	testdb	PRIMARY	testdb	s_emp	PRIMARY KEY	YES
def	testdb	s_emp_ibfk_1	testdb	s_emp	FOREIGN KEY	YES
def	testdb	s_emp_ibfk_2	testdb	s_emp	FOREIGN KEY	YES

ADD AND REMOVE DATA CONSTRAINTS

Example

s_emp 테이블의 외래키 삭제 및 추가

```
SQL> ALTER TABLE s_emp
DROP FOREIGN KEY s_emp_ibfk_1;
Table 's_emp' altered.
```

```
SQL> ALTER TABLE s_emp
ADD CONSTRAINT s_emp_fk_id FOREIGN KEY (mgr)
REFERENCES s_emp(empno);
```

```
mysql> select * from information_schema.table_constraints where table_name='s_emp';
```

CONSTRAINT_CATALOG	CONSTRAINT_SCHEMA	CONSTRAINT_NAME	TABLE_SCHEMA	TABLE_NAME	CONSTRAINT_TYPE	ENFORCED
def	testdb	PRIMARY	testdb	s_emp	PRIMARY KEY	YES
def	testdb	s_emp_fk_id	testdb	s_emp	FOREIGN KEY	YES
def	testdb	s_emp_ibfk_2	testdb	s_emp	FOREIGN KEY	YES

```
3 rows in set (0.01 sec)
```

DROP A TABLE

DROP TABLE 명령어를 사용해 데이터베이스에서 테이블 제거.
DROP TABLE 명령어는 한번 실행되면 취소할 수 없다.

Syntax

```
DROP TABLE table_name
```

Example

S_REGION 테이블 삭제

```
SQL> DROP TABLE s_region;
```

Table 'S_REGION' dropped.

Simplify Data Access with Views

- 테이블 뷰를 만들어 논리적 하위 집합 또는 데이터 조합을 표시한다.
- 뷰는 실제 데이터가 포함되지 않는다.
- 뷰 이름은 테이블과 같은 네임스페이스를 사용하므로 스키마 내 다른 이름과 중복되면 안된다.

- 장점
 - 접근 제어로 보안 제공
 - 데이터 관리가 편리

- 단점
 - 삽입, 삭제, 갱신 연산에 제약이 있다.

CREATE VIEWS

Syntax

```
CREATE VIEW view_name [ (alias, [alias]...)  
AS query  
WHERE  
WITH CHECK OPTION
```

where	<i>view_name</i>	뷰 이름
	<i>alias</i>	별칭
	<i>query</i>	SELECT 문
	<i>WITH CHECK OPTION</i>	해당 옵션을 사용하면 INSERT/UPDATE 가능
	<i>WITH READ ONLY</i>	읽기 전용 뷰
	<i>constraint</i>	CHECK OPTION에 할당 된 제약조건

CREATE VIEWS

Example

부서 번호가 10인 직원 번호, 이름, 직무가 포함된 뷰를 생성

```
SQL> CREATE VIEW empvu10
  2 AS SELECT empno, ename, job
  3 FROM s_emp
  4 WHERE deptno = 10;
```

View 'EMPVU10' created.

```
mysql> select * from empvu10;
+-----+-----+-----+
| empno | ename  | job      |
+-----+-----+-----+
| 7782  | CLARK  | MANAGER  |
| 7839  | KING   | PRESIDENT|
| 7934  | MILLER | CLERK    |
+-----+-----+-----+
3 rows in set (0.00 sec)
```

CREATE VIEWS

Example

부서 번호가 20인 뷰를 생성. 직원 번호는 ID, 이름은 EMPLOYEE, 직무는 TITLE로 표현.

```
SQL> CREATE VIEW empvu20 (id, employee, title)
  2 AS SELECT empno, ename, job
  3 FROM s_emp
  4 WHERE deptno = 20;
```

View 'EMPVU20' created.

```
SQL> SELECT *
  2 FROM empvu20;
```

ID	EMPLOYEE	TITLE
7369	SMITH	CLERK
7566	JONES	MANAGER
7788	SCOTT	ANALYST
7876	ADAMS	CLERK
7902	FORD	ANALYST

5 rows selected.

CREATE VIEWS

Example

월급이 1500 이상인 뷰를 생성. 직원 번호는 ID, 이름은 NAME, 월급은 MONTHLY_SALARY로 표현.

```
SQL> CREATE VIEW salvu1500
  AS SELECT empno ID, ename NAME, sal MONTHLY_SALARY
  FROM s_emp
  WHERE sal >= 1500;
```

View 'SALVU1500' created.

```
SQL> SELECT *
  2 FROM salvu1500;
```

ID	NAME	MONTHLY_SALARY
7499	ALLEN	1600
7566	JONES	2975
7698	BLAKE	2850
7782	CLARK	2450
7788	SCOTT	3000
7839	KING	5000
7844	TURNER	1500
7902	FORD	3000

8 rows selected.

CREATE VIEWS

Example

s_emp 테이블에서 부서 번호가 30인 뷰를 'WITH CHECK OPTION'을 이용하여 생성

```
SQL> CREATE VIEW empvu30
2 AS SELECT *
3 FROM s_emp
4 WHERE deptno = 30
5 WITH CHECK OPTION;
```

부서 번호를 20으로 업데이트

```
UPDATE empvu30
SET deptno=20
WHERE deptno=30;
TBR-10010: Statement does not satisfy the WHERE clause of the view.
```

부서 번호가 30인 새로운 사원 정보 입력

```
SQL> INSERT INTO empvu30
VALUES (9999, 'TABU', 'STUDENT', NULL, '22-10-05', 1500, NULL, 30);

1 row inserted.
```

CONFIRM VIEW NAMES AND STRUCTURES

현재 사용자에게 속한 뷰의 정보를 조회할 수 있다.

Example

```
SQL> SHOW FULL TABLES IN DB명  
WHERE TABLE_TYPE LIKE 'VIEW';
```

Example

```
mysql> show full tables in testdb where table_type like 'VIEW';  
+-----+-----+  
| Tables_in_testdb | Table_type |  
+-----+-----+  
| empvu10          | VIEW      |  
+-----+-----+  
1 row in set (0.00 sec)
```

DROP A VIEW

Syntax

```
DROP VIEW view_name
```

Example

EMPVU10 뷰 삭제

```
SQL> DROP VIEW empvu10;
```

View 'EMPVU10' dropped.

데이터 조작용어(Data Manipulation Language)

데이터 조작용어(이하 DML)는 데이터베이스에 저장된 데이터에 대한 질의, 삽입, 갱신, 삭제를 수행하기 위한 SQL 문장

명령어	설명
SELECT	데이터를 조회한다.
INSERT	데이터를 삽입한다.
UPDATE	데이터를 변경한다.
DELETE	데이터를 삭제한다.

DISPLAY TABLE STRUCTURE

칼럼의 이름과 데이터 타입을 포함한 테이블 구조 확인

Syntax

```
DESC[RIBE] tablename
```

Example

```
SQL> DESCRIBE s_emp;
```

COLUMN_NAME	TYPE	CONSTRAINT
EMPNO	NUMBER(4)	PRIMARY KEY
ENAME	VARCHAR(10)	
JOB	VARCHAR(9)	
MGR	NUMBER(4)	
HIREDATE	DATE	
SAL	NUMBER(7,2)	
COMM	NUMBER(7,2)	
DEPTNO	NUMBER(2)	REFERENTIAL

INDEX_NAME	TYPE	COLUMN_NAME
PK_EMP	NORMAL	EMPNO

Display Data with the SELECT Statement

SQL SELECT 문을 사용하여 데이터베이스 테이블의 데이터를 조회할 수 있다.

Syntax

```
SELECT column_information
FROM table(s)
WHERE condition
ORDER BY expression or keyword
```

where	SELECT	검색할 열, 식 또는 상수를 지정
	FROM	데이터를 가지고 올 테이블 지정
	WHERE	특정 행을 검색할 기준(선택사항)
	ORDER BY	조회된 데이터를 정렬

DISPLAY ALL DATA IN A TABLE

테이블의 모든 칼럼을 출력하려면 SELECT 키워드에 '*'를 입력

Syntax

```
SELECT *
FROM table_name
```

Example

```
SQL> SELECT *
      2 FROM s_dept;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

4 rows selected.

DISPLAY ALL DATA IN A TABLE

조회하고 싶은 칼럼 이름과 해당 칼럼이 정의된 테이블을 입력하여 조회

Syntax

```
SELECT [DISTINCT] column_name [, column_name]  
FROM table_name
```

Example

```
SQL> SELECT dname  
2 FROM s_dept;
```

```
DNAME  
-----  
ACCOUNTING  
RESEARCH  
SALES  
OPERATIONS
```

4 rows selected.

DISPLAY ALL DATA IN A TABLE

S_EMP 테이블의 칼럼과 구조를 조회하고, 사원 이름, 급여를 출력

Example

```
SQL> SELECT dname
2 FROM s_dept;
```

```
SQL> DESC s_emp;
```

```
SQL> SELECT ename, sal
2 FROM s_emp;
```

ENAME	SAL
SMITH	800
ALLEN	1600
WARD	1250
JONES	2975
MARTIN	1250
BLAKE	2850
CLARK	2450
SCOTT	3000
KING	5000
TURNER	1500
ADAMS	1100
JAMES	950
FORD	3000

DISPLAY UNIQUE COLUMNS OF DATA

DISTINCT 절을 사용하여 고유한 데이터 행을 조회한다. **DISTINCT** 절은 **SELECT** 문 결과가 반환되기 전 중복된 행을 제거한다.

Syntax

```
SELECT [DISTINCT] column_name [, column_name]  
FROM table_name
```

Example

```
SQL> SELECT job  
2 FROM s_emp;  
JOB  
-----  
CLERK  
SALESMAN  
SALESMAN  
MANAGER  
SALESMAN  
MANAGER  
MANAGER  
ANALYST  
PRESIDENT  
SALESMAN  
CLERK  
CLERK  
ANALYST  
STUDENT
```

```
SQL> SELECT DISTINCT job  
2 FROM s_emp;  
  
JOB  
-----  
CLERK  
ANALYST  
STUDENT  
PRESIDENT  
SALESMAN  
MANAGER
```

DISPLAY SPECIFIC COLUMN NAMES

별칭 - SELECT 문에서 칼럼 이름을 대체로 정의할 수 있다.

별칭에 공백, 특수문자가 포함되어 있거나, 대소문자 구분이 필요하면 별칭을 큰 따옴표로 묶는다.

Example

```
SQL> SELECT DISTINCT job "Title"
2 FROM s_emp;
```

```
Title
-----
CLERK
ANALYST
STUDENT
PRESIDENT
SALESMAN
MANAGER
```

```
SQL> SELECT ename EMPLOYEES
2 FROM s_emp;
```

```
EMPLOYEES
-----
SMITH
ALLEN
WARD
JONES
MARTIN
BLAKE
CLARK
SCOTT
KING
TURNER
ADAMS
JAMES
FORD
```

DISPLAY SPECIFIC ROWS OF DATA

WHERE절을 사용해 조건에 맞는 행을 조회한다.

Syntax

```
SELECT { * | column_name [, column_name...] }
FROM table_name
WHERE condition
```

Example

```
SQL> SELECT ename
2    FROM s_emp
3    WHERE sal > 2000;
```

```
ENAME
-----
KING
BLAKE
CLARK
JONES
FORD
SCOTT
```

6 rows selected.

DISPLAY SPECIFIC ROWS OF DATA

Comparison Operators Overview

=	Equal to
<>	Not equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

DISPLAY SPECIFIC ROWS OF DATA

Comparison Operators Overview

BETWEEN...AND...	between two values
NOT BETWEEN... AND...	Not between two values
IN (list)	Equal to any member of the following list
NOT IN (list)	Not Equal to any member of the following list
LIKE	Match a character pattern using wildcard characters
IS NULL	Is a null
IN NOT NULL	Is not a null

DISPLAY SPECIFIC COLUMN NAMES

=

부서 번호가 20인 직원의 부서 번호, 사원 이름, 직무를 조회

Example

```
SQL> SELECT deptno, ename, job
2   FROM s_emp
3   WHERE deptno = 20;
```

DEPTNO	ENAME	JOB
20	JONES	MANAGER
20	FORD	ANALYST
20	SMITH	CLERK
20	SCOTT	ANALYST
20	ADAMS	CLERK

DISPLAY SPECIFIC COLUMN NAMES

=

BLAKE의 사원 번호, 이름, 직무, 급여를 조회

Example

```
SQL> SELECT empno, ename, job, sal
2 FROM s_emp
3 WHERE ename = 'BLAKE';
```

```
mysql> select empno, ename, job from s_emp where ename = 'BLAKE';
+-----+-----+-----+
| empno | ename | job   |
+-----+-----+-----+
| 7698  | BLAKE | MANAGER |
+-----+-----+-----+
1 row in set (0.00 sec)
```

DISPLAY SPECIFIC COLUMN NAMES

=

입사 날짜가 82/12/22일인 사원의 이름, 부서 번호, 입사 날짜 조회

Example

```
SQL> SELECT ename, deptno, hiredate
2  FROM s_emp
3  WHERE hiredate = '82/12/22;
```

ENAME	DEPTNO	HIREDATE
SCOTT	20 0082/12/22	

DISPLAY SPECIFIC COLUMN NAMES



직무가 **MANAGER**가 아닌 사원의 이름, 직무, 급여 조회

Example

```
SQL> SELECT ename, job, sal
2   FROM s_emp
3   WHERE job <> 'MANAGER';
```

ENAME	JOB	SAL
KING	PRESIDENT	5000
MARTIN	SALESMAN	1250
ALLEN	SALESMAN	1600
TURNER	SALESMAN	1500
JAMES	CLERK	950
WARD	SALESMAN	1250
FORD	ANALYST	3000
SMITH	CLERK	800
SCOTT	ANALYST	3000
ADAMS	CLERK	1100
MILLER	CLERK	1300

11 rows selected.

DISPLAY ROWS WITH COMPLEX CONDITIONS

OR

부서 번호가 20이거나, 직무가 SALESMAN인 사원의 이름, 급여, 부서 번호, 직무 조회

Example

```
SQL> SELECT ename, sal, deptno, job
2 FROM s_emp
3 WHERE deptno = 20
4 OR job = 'SALESMAN';
```

ENAME	SAL	DEPTNO	JOB
JONES	2975	20	MANAGER
MARTIN	1250	30	SALESMAN
ALLEN	1600	30	SALESMAN
TURNER	1500	30	SALESMAN
WARD	1250	30	SALESMAN
FORD	3000	20	ANALYST
SMITH	800	20	CLERK
SCOTT	3000	20	ANALYST
ADAMS	1100	20	CLERK

9 rows selected.

DISPLAY ROWS WITH COMPLEX CONDITIONS

우선 순위는 모든 비교 연산자, **AND** 그리고 **OR** 순서다.

Rules of Precedence

ORDER Evaluated	Operator
1	All Comparison Operators (=, <>, >, >=, <, <=, IN, LIKE, IN NULL, BETWEEN...AND)
2	AND
3	OR

DISPLAY ROWS WITH COMPLEX CONDITIONS

급여가 1,000 이상이고, 직무 번호가 10 또는 20인 사원의 이름, 급여, 직무 번호 조회

Example

```
SQL> SELECT ename, sal, deptno
      FROM s_emp
      WHERE sal >= 1000 AND deptno = 10 OR deptno = 20;
```

ENAME	SAL	DEPTNO
SMITH	800	20
JONES	2975	20
CLARK	2450	10
SCOTT	3000	20
KING	5000	10
ADAMS	1100	20
FORD	3000	20

7 rows selected.

DISPLAY ROWS WITH COMPLEX CONDITIONS

Example

```
SQL> SELECT ename, sal, deptno  
2 FROM s_emp  
3 WHERE sal >= 1000 AND (deptno = 10 OR deptno = 20);
```

ENAME	SAL	DEPTNO
KING	5000	10
CLARK	2450	10
JONES	2975	20
FORD	3000	20
SCOTT	3000	20
ADAMS	1100	20
MILLER	1300	10

7 rows selected.

ORDER THE ROWS DISPLAYED

특정 칼럼을 기준으로 정렬이 가능하다.

Syntax

```
SELECT { * | column_name [, column_name...] }
FROM table_name
WHERE condition
ORDER BY column_name {ASC | DESC} [, column_name [ASC | DESC] ...]
```

where	column_name	칼럼 이름
	table_name	테이블 이름
	ASC	행을 오름차순으로 정렬(Default)
	DESC	행을 내림차순으로 정렬

ORDER THE ROWS DISPLAYED

급여를 기준으로 오름차순 정렬

Example

```
SQL> SELECT ename, sal  
2 FROM s_emp  
3 WHERE deptno = 30  
4 ORDER BY sal;
```

ENAME	SAL
JAMES	950
MARTIN	1250
WARD	1250
TURNER	1500
ALLEN	1600
BLAKE	2850

6 rows selected.

ORDER THE ROWS DISPLAYED

급여를 기준으로 내림차순 정렬

Example

```
SQL> SELECT ename, sal  
2 FROM s_emp  
3 WHERE deptno = 30  
4 ORDER BY sal DESC;
```

ENAME	SAL
BLAKE	2850
ALLEN	1600
TURNER	1500
MARTIN	1250
WARD	1250
JAMES	950

6 rows selected.