

db

🕒 작성일시	@2022년 8월 1일 오전 9:41
▼ 강의 번호	
📅 Date	@2022년 8월 1일
🔗 자료	
☑ 복습	<input type="checkbox"/>

전체 데이터베이스 목록 보기 : `show databases;`

```
MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
4 rows in set (0.001 sec)

MariaDB [(none)]>
```

데이터베이스 생성하기 : `create database 생성이름;`

```

MariaDB [(none)]> create database test;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
| test |
+-----+
5 rows in set (0.001 sec)

MariaDB [(none)]> _

```

사용할 데이터 베이스 선택 : use 생성된 이름;

```

MariaDB [(none)]> use test;
Database changed
MariaDB [test]> _

```

- 테이블 생성 하기. :

```

CREATE TABLE 테이블이름(

                                name      varchar(5),

                                sno        int,

                                dept       varchar(10),

                                primary    key(sno)

) ;

```

```
MariaDB [test]> CREATE TABLE student(  
-> name      varchar(5),  
-> sno       int,  
-> dept      varchar(10),  
-> primary key(sno) );  
Query OK, 0 rows affected (0.012 sec)
```

자료형

1. 문자 데이터

- char 고정길이 : 8글자데이터를 입력할때 나머지글자는 모두 공백으로 채워서 저장
- varchar 가변길이 : 8글자 데이터를 입력할때 8 글자만 저장된다. 주로 사용하는 문자 자료형

```
char(20); /* fixed-length*/  
varchar(20); /*variable-length*/
```

2. 텍스트 데이터

긴 문자열을 저장할때 사용한다.

- text
- tinytext : 작은 문자(msql만 사용가능)

3. 숫자 데이터

- int 주로 사용하는 숫자 자료형
- tinyint
- smallint
- bigint

4. 날짜데이터

- timestamp - 현재 날짜와, 시간을 자동입력

- datetime - 날짜와 시간

테이블 작성

1. 설계(디자인)

테이블에 저장할 적절한 항목들과 그 항목들을 저장할 데이터 형과 크기를 설계
이름, 주소, 전화번호, 성, 음식....

2. 정제

테이블 작성시에는 기본키 설정이 중요(PRIMARY KEY)

이름의 경우에는 성과 이름으로 분리

주소의 경우에는 하나의 필드로 저장하는 것보다. 예를 들면 시,군,구 별로 따로 분리

3. SQL 구문 생성

```
-  
CREATE TABLE 테이블이름(  
                                name      varchar(5),  
                                sno        int,  
                                dept       varchar(10),  
                                primary    key(sno)  
);
```

```
CREATE TABLE person(  
person_id SMALLINT UNSIGNED,  
fname VARCHAR(20),  
lname VARCHAR(20),  
gender CHAR(1),  
birth_date DATE,
```

```
street VARCHAR(30),  
city VARCHAR(20),  
country VARCHAR(20),  
postal_code VARCHAR(20),  
CONSTRAINT pk_person PRIMARY KEY (person_id));
```

교재 예제 설치

<https://dev.mysql.com/doc/index-other.html>

<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/4f9de4c0-aa29-4a52-b863-97da05d053dd/sakila-db.zip>

<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/2339a2be-1eea-4072-a9bd-22e857da8336/LearningSQLExample.sql>

```
mysql> SOURCE C:/temp/sakila-schema.sql;  
mysql> SOURCE C:/temp/sakila-data.sql;
```

c:/안에 temp라는 폴더 생성후

sakila-data

sakila-schema 파일 이동후

```
mysql> SOURCE C:/temp/sakila-schema.sql;  
mysql> SOURCE C:/temp/sakila-data.sql;
```

입력

use test;로 넘어 간뒤 show tables; 사용하여 만들어진 테이블 확인

use sakila; 에서 테이블 생성하기

```
mysql> CREATE TABLE favorite_food
-> (person_id SMALLINT UNSIGNED,
->   food VARCHAR(20),
->   CONSTRAINT pk_favorite_food PRIMARY KEY (person_id, food),
->   CONSTRAINT fk_fav_food_person_id FOREIGN KEY (person_id)
->     REFERENCES person (person_id)
-> );
Query OK, 0 rows affected (0.10 sec)
```

sakila 안에 person table이 있어야 함

```
CREATE TABLE favorite_food(
person_id SMALLINT UNSIGNED,
food VARCHAR(20),
CONSTRAINT pk_favorite_food PRIMARY KEY (person_id, food),
CONSTRAINT fk_fav_food_person_id FOREIGN KEY (person_id)
REFERENCES person (person_id));
```

다 만들면 DESC person; 조회 , DESC favorite_food; 조회

테이블 수정

1. 데이터 삽입

- 데이터를 추가할 테이블 이름
- 데이터를 추가할 테이블의 열이름

- 열에 넣을 값

```
INSERT INTO person
(person_id, fname, lname, birth_date)
VALUES (0, 'William','Turner', '1972-05-27');
```

person 테이블의 person_id 열을 자동증가하는 형태로 변경

```
ALTER TABLE person MODIFY person_id SMALLINT UNSIGNED
AUTO_INCREMENT;
```

person 테이블에서 person_id, fname, lname, birth_date 필드값들 전체를 조회

```
SELECT person_id, fname, lname, eye_color, birth_date FROM person;
SELECT * FROM person; - - 전체 필드를 다 적지 않고 * 로 전체 필드를 표시
```

person 테이블에서 person_id = 1인 조건에 해당하는 person_id, fname, lname, birth_date 필드값들을 조회

```
SELECT person_id, fname, lname, birth_date
FROM person
WHERE person_id = 1;
```

전체주석 /* □□□ */

```
WHERE person_id = 1; -- 한줄 주석
```

```
SELECT * FROM person;

SELECT person_id, fname, lname, birth_data
FROM person
WHERE person_id = 1;

//favorite_food라는 테이블에 입력
INSERT INTO favorite_food (person_id, food)
```

```

VALUES(1, 'pizza');
INSERT INTO favorite_food (person_id, food)
VALUES(1, 'cookies');
INSERT INTO favorite_food (person_id, food)
VALUES(1, 'nachos');

//food 열을 favorite_food라는 테이블로부터 조건은 person_id=1인 값만 순서를 food열을
//오름차순으로 정렬하여 조회
SELECT food
FROM favorite_food
WHERE person_id = 1
ORDER BY food;

INSERT INTO person
(person_id, fname, lname, birth_date,
street, city, state, country, postal_code)
VALUES (2, 'Susan', 'Smith', '1975-11-02',
'23 Maple St.', 'Arlington', 'VA', 'USA', '20220');

UPDATE person
SET street = '1225 Tremont St.',
city = 'Boston',
state = 'MA',
country = 'USA',
postal_code = '02138'
WHERE person_id = 1;

//person_id가 2인 레코드를 삭제
DELETE FROM person
WHERE person_id = 2;

```

자주 등장하는 에러

```

mysql> INSERT INTO person
-> (person_id, fname, lname, gender, birth_date)
-> VALUES (1, 'Charles', 'Fulton', 'M', '1968-01-15');
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'

```

- 고유 키값이 중복된 데이터를 입력하려고 시도할 때 에러 발생


```
mysql> INSERT INTO favorite_food (person_id, food)
-> VALUES (999, 'lasagna');
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint
fails ('bank'. 'favorite_food', CONSTRAINT 'fk_fav_food_person_id' FOREIGN KEY
('person_id') REFERENCES 'person' ('person_id'))
```

- 존재하지 않는 왜래 키 foreign key 를 참조할때 에러 발생

```
mysql> UPDATE person
-> SET gender = 'Z'
-> WHERE person_id = 1;
ERROR 1265 (01000): Data truncated for column 'gender' at row 1
```

- 열 값 위반, : 선언한 데이터형을 벗어났을때 에러 발생

```
mysql> UPDATE person
-> SET birth_date = 'DEC-21-1980'
-> WHERE person_id = 1;
ERROR 1292 (22007): Incorrect date value: 'DEC-21-1980' for column 'birth_date'
at row 1
```

- 잘못된 날짜 변환 : 날짜의 기본형인 년*월*일 로 입력되지 않고 월*일*년으로 입력되어 에러 발생

```
mysql> DROP TABLE favorite_food;
Query OK, 0 rows affected (0.56 sec)
mysql> DROP TABLE person;
Query OK, 0 rows affected (0.05 sec)
```

- 테이블 제거

DROP TABLE 테이블이름;

DESC customer; 구조를 자세히 보기

MySQL에 Client에서 use test; 들어간뒤

SOURCE C:/temp/LearningSQLExample.sql;

```
SELECT emp_id, fname, lname  
FROM employee  
WHERE lname = 'Bkadfl';
```

```
SELECT fname, lname  
FROM employee;
```

쿼리에 사용되는 구문들

Select	쿼리 결과에 포함 시킬 열들 결정
FROM	결과를 검색할 테이블, 테이블들을 조인하는 방법 등
WHERE	원하지 않는 데이터를 걸러내는 조건 설정
GROUP BY	공통열 값을 기준으로 행들을 그룹화
HAVING	원하지 않는 그룹을 걸러내는 조건 설정
ORDER BY	하나 또는 하나 이상의 열들을 기준으로 최종 결과의 행들을 정렬

1. SELECT

select 절을 완전하게 이해하려면 from절을 먼저 이해해야 한다.

```
mysql> SELECT *
-> FROM department;
+-----+-----+
| dept_id | name          |
+-----+-----+
| 1       | Operations    |
| 2       | Loans         |
| 3       | Administration |
+-----+-----+
3 rows in set (0.04 sec)
```



SELECT * FROM department;

이 쿼리에서의 FROM은 department이라는 하나의 테이블의 모든 열의 결과에 포함하는 것을 나타낸다. *asterisk 문자는 모든 열을 지정한다.

```
mysql> SELECT dept_id, name
-> FROM department;
+-----+-----+
| dept_id | name          |
+-----+-----+
| 1       | Operations    |
| 2       | Loans         |
| 3       | Administration |
+-----+-----+
3 rows in set (0.01 sec)
```

또는 하나의 열만 선택하여 결과를 볼수 도 있다.

```
mysql> SELECT name
-> FROM department;
+-----+
| name          |
+-----+
| Operations    |
| Loans         |
| Administration |
+-----+
3 rows in set (0.00 sec)
```

숫자나 문자를 그냥 출력

기존열의 값을 계산한 결과를 출력

함수를 사용한 결과

```
SELECT emp_id,  
       'ACTIVE',  
       emp_id * 3.14159,  
       UPPER(lname)  
FROM employee;
```

```
SELECT VERSION(),  
       USER(),  
       DATABASE()
```

VERSION()	USER()	DATABASE()
10.8.3-MariaDB	root@localhost	bank

컬럼 별칭

```
SELECT emp_id,  
       'ACTIVE' AS STATUS,  
       emp_id * 3.14159 AS empid_x_pi,  
       UPPER(lname) AS last_name_upper  
FROM employee;
```

```
SELECT emp_id,  
       'ACTIVE' AS STATUS,  
       emp_id * 3.14159 AS 상태,  
       UPPER(lname) AS 대문자성  
FROM employee;
```

```
SELECT emp_id 사번,  
'ACTIVE' 상태,  
UPPER(lname) 대문자성  
FROM employee;
```

중복 제거 DISTINCT

상황에 따라 쿼리가 중복된 행을 반환할수 있다. 고유한 하나의 값만 남기고 나머지는 제거한 값을 확인 할수 있다.

```
mysql> SELECT cust_id  
-> FROM account;
```

```
mysql> SELECT DISTINCT cust_id  
-> FROM account;
```

```
SELECT cust_id  
FROM account;
```

```
SELECT DISTINCT cust_id  
FROM account;
```

FROM 절

지금까지는 from 절에 단 하나의 테이블만 지정하였는데 대부분의 실제 SQL 구문에서는 하나 이상의 테이블을 목록으로 정의하여 사용된다.

FROM 절은 쿼리에 사용되는 테이블을 명시할 뿐만아니라 테이블들을 서로 연결하는 수단도 정의하게 된다.

Permanent Table 영구테이블 create table 로 생성된 테이블

Temporary Table 임시테이블 서브 쿼리로 반환된 행들, 메모리에 임시 저장된 휘발성 테이블

Virtual Table 가상테이블 create view 로 생성된 테이블

파생테이블 (← 임시테이블) Temporary Table

```
mysql> SELECT e.emp_id, e.fname, e.lname  
-> FROM (SELECT emp_id, fname, lname, start_date, title  
->        FROM employee) e;
```

```
SELECT e.emp_id, e.fname, e.lname  
FROM (SELECT emp_id, fname, lname, start_date, title  
FROM employee)e;
```

Virtual Table 가상테이블

```
mysql> CREATE VIEW employee_vw AS  
-> SELECT emp_id, fname, lname,  
->        YEAR(start_date) start_year  
-> FROM employee;
```

```
+ :: mysql> SELECT emp_id, start_year  
-> FROM employee_vw;
```

```
CREATE VIEW employee_vw AS  
SELECT emp_id, fname, lname,  
YEAR(start_date) start_year  
FROM employee;
```

```
SELECT emp_id, start_year  
FROM employee_vw;
```

Page 51 테이블 연결

```
mysql> SELECT employee.emp_id, employee.fname,  
->        employee.lname, department.name dept_name  
-> FROM employee INNER JOIN department  
-> ON employee.dept_id = department.dept_id;
```

```

SELECT e.emp_id, e.fname, e.lname,
       d.name dept_name
FROM employee e INNER JOIN department d
ON e.dept_id = d.dept_id;

```

```

SELECT e.emp_id, e.fname, e.lname,
       d.name dept_name
FROM employee AS e INNER JOIN department AS d
ON e.dept_id = d.dept_id;

```

```

SELECT employee.emp_id, employee.fname,
       employee.lname, department.name dept_name
FROM employee INNER JOIN department
ON employee.dept_id = department.dept_id;

SELECT e.emp_id, e.fname, e.lname,
       d.name dept_name
FROM employee e INNER JOIN department d
ON e.dept_id = d.dept_id;

SELECT e.emp_id, e.fname, e.lname,
       d.name dept_name
FROM employee AS e INNER JOIN department AS d
ON e.dept_id = d.dept_id;

```

page 52, WHERE 절

where절은 결과에 출력되기를 원하지 않는 행을 걸러내는 방법이다.

```
mysql> SELECT emp_id, fname, lname, start_date, title
-> FROM employee
-> WHERE title = 'Head Teller';
```

emp_id	fname	lname	start_date	title
6	Helen	Fleming	2008-03-17	Head Teller
10	Paula	Roberts	2006-07-27	Head Teller
13	John	Blake	2004-05-11	Head Teller
16	Theresa	Markham	2005-03-15	Head Teller

```
SELECT emp_id, fname, lname, start_date, title
FROM employee
WHERE title = 'Head Teller';
```

조건 2개를 동시에 만족하는 데이터 출력

```
mysql> SELECT emp_id, fname, lname, start_date, title
-> FROM employee
-> WHERE title = 'Head Teller'
-> AND start_date > '2006-01-01';
```

```
SELECT emp_id, fname, lname, start_date, title
FROM employee
WHERE title = 'Head Teller';
AND start_date > '2006-01-01';
```

Group by절과 Having 절

```
mysql> SELECT d.name, count(e.emp_id) num_employees
-> FROM department d INNER JOIN employee e
-> ON d.dept_id = e.dept_id
-> GROUP BY d.name
-> HAVING count(e.emp_id) > 2;
```


GROUP BY 열을 기준으로 행들의 값으로 그룹으로 나누고 그 나뉜 그룹에 조건을 적용하는 것이 HAVING 이다.

```
SELECT d.name, count(e.emp_id) num_employees
FROM department d INNER JOIN employee e
ON d.dept_id = e.dept_id
GROUP BY d.name
HAVING count(e.emp_id) > 2;
```

ORDER BY 절

일반적으로 쿼리는 반환된 결과셋의 행은 특정한 순서로 정렬되지는 않는다. 결과를 원하는 순서로 정렬하려면 ORDER BY 절을 사용한다.

```
mysql> SELECT open_emp_id, product_cd
-> FROM account;
+-----+-----+
| open_emp_id | product_cd |
+-----+-----+
|          10 | CHK        |
|          10 | SAV        |
|          10 | CD         |
|          10 | CHK        |
|          10 | SAV        |
|          13 | CHK        |
|          13 | MM         |
|           1 | CHK        |
|           1 | SAV        |
|           1 | MM         |
|          16 | CHK        |
|           1 | CHK        |
|           1 | CD         |
+-----+-----+
```

```
mysql> SELECT open_emp_id, product_cd
-> FROM account
-> ORDER BY open_emp_id;
+-----+-----+
| open_emp_id | product_cd |
+-----+-----+
|           1 | CHK        |
|           1 | SAV        |
|           1 | MM         |
|           1 | CHK        |
|           1 | CD         |
|           1 | CHK        |
|           1 | MM         |
|           1 | CD         |
|          10 | CHK        |
|          10 | SAV        |
|          10 | CD         |
|          10 | CHK        |
|          10 | SAV        |
+-----+-----+
```

```
SELECT open_emp_id, product_cd
FROM account;
```

```
SELECT open_emp_id, product_cd
```

```
FROM account
ORDER BY open_emp_id;
```

정렬의 기준이 여러개인 경우는 첫번째 정렬을 마친 값들중 동일한 값들만 만 다시한번 정렬

```
mysql> SELECT open_emp_id, product_cd
-> FROM account
-> ORDER BY open_emp_id, product_cd;
```

open_emp_id	product_cd
1	CD
1	CD
1	CHK
1	CHK
1	CHK
1	MM
1	MM
1	SAV
10	BUS
10	CD
10	CD
10	CHK

```
SELECT open_emp_id, product_cd
FROM account
ORDER BY open_emp_id, product_cd;
```

정렬의 기본은 오름차순으로 적시 하지 않으면 오름차순 정렬되고
DESC 를 적으면 내림차순으로 정렬된다.

```
mysql> SELECT account_id, product_cd, open_date, avail_balance
-> FROM account
-> ORDER BY avail_balance DESC;
```

account_id	product_cd	open_date	avail_balance
29	SBL	2004-02-22	50000.00
28	CHK	2003-07-30	38552.05
24	CHK	2002-09-30	23575.12
15	CD	2004-12-28	10000.00
27	BUS	2004-03-22	9345.55

```
SELECT account_id, product_cd, open_date, avail_balance
FROM account
ORDER BY avail_balance DESC;
```

```
alter session set nls_date_format='RR/MM/DD';
```

```
drop table emp;
drop table dept;
```

```
CREATE TABLE DEPT
(DEPTNO number(10),
DNAME VARCHAR2(14),
LOC VARCHAR2(13) );
```

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

```
CREATE TABLE EMP (
EMPNO          NUMBER(4) NOT NULL,
ENAME          VARCHAR2(10),
JOB            VARCHAR2(9),
MGR            NUMBER(4) ,
HIREDATE       DATE,
SAL            NUMBER(7,2),
COMM           NUMBER(7,2),
DEPTNO         NUMBER(2) );
```

```
INSERT INTO EMP VALUES (7839, 'KING', 'PRESIDENT', NULL, '81-11-17', 5000, NULL, 10);
INSERT INTO EMP VALUES (7698, 'BLAKE', 'MANAGER', 7839, '81-05-01', 2850, NULL, 30);
```

```

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

commit;

drop table salgrade;

create table salgrade
( grade    number(10),
  losal    number(10),
  hisal    number(10) );

insert into salgrade values(1,700,1200);
insert into salgrade values(2,1201,1400);
insert into salgrade values(3,1401,2000);
insert into salgrade values(4,2001,3000);
insert into salgrade values(5,3001,9999);

commit;

```

Quiz 001

사원 테이블에서 사원 번호와 이름과 월급을 출력해 보겠습니다.

```

/*
SELECT EMPNO, ENAME, SAL
FROM EMP;

SELECT empno, ename, sal
FROM emp;

SELECT empno, ename, sal FROM emp;

SELECT empno, ename, sal
FROM emp;
*/

SELECT empno, ename, sal
FROM emp;

```

emp (14r x 3c)		
EMPNO	ENAME	SAL
7,839	KING	5,000
7,698	BLAKE	2,850
7,782	CLARK	2,450
7,566	JONES	2,975
7,654	MARTIN	1,250

Quiz.
002

사원 테이블을 모든 열(column)들을 전부 출력해 보겠습니다.

```
-- SELECT *
-- FROM emp;

SELECT EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM,
FROM emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7,839	KING	PRESIDENT	(NULL)	1981-11-17	5,000	(NULL)	10
7,698	BLAKE	MANAGER	7,839	1981-05-01	2,850	(NULL)	30
7,782	CLARK	MANAGER	7,839	1981-05-09	2,450	(NULL)	10
7,566	JONES	MANAGER	7,839	1981-04-01	2,975	(NULL)	20
7,654	MARTIN	SALESMAN	7,698	1981-09-10	1,250	1,400	30
7,499	ALLEN	SALESMAN	7,698	1981-02-11	1,600	300	30
7,844	TURNER	SALESMAN	7,698	1981-08-21	1,500	0	30
7,900	JAMES	CLERK	7,698	1981-12-11	950	(NULL)	30
7,521	WARD	SALESMAN	7,698	1981-02-23	1,250	500	30
7,902	FORD	ANALYST	7,566	1981-12-11	3,000	(NULL)	20
7,369	SMITH	CLERK	7,902	1980-12-11	800	(NULL)	20

Quiz 003

사원 테이블의 사원 번호와 이름과 월급을 출력하는데 컬럼명을 한글로 '사원 번호', '사원 이름'으로 출력해 보겠습니다.

출력 결과

사원 번호	이름	Salary
7839	KING	5000
7698	BLAKE	2850
7782	CLARK	2450
7566	JONES	2975
:	:	:

1	SELECT EMPNO AS 사 원 번 호 , ENAME AS 이 름 , SAL AS '사 원 Salary'
2	FROM emp;

emp (14r x 3c)		
사원번호	이름	사원 Salary
7,839	KING	5,000
7,698	BLAKE	2,850

Quiz 004

사원 테이블의 이름과 월급을 서로 붙여서 출력해 보겠습니다.

KING5000
BLAKE2850
CLARK2450
JONES2975
...

오스, 127.0.0.1 데이터베이스 - Dal

```

1 SELECT CONCAT(ENAME, SAL)
2 FROM emp;
3

```

emp (14r x 1c)

CONCAT(ENAME, SAL)
KING5000
BLAKE2850
CLARK2450
JONES2975
MARTIN1250
ALLEN1600
TURNER1500

× 필터: 정규 표현식

Quiz 005

직업정보
KING 의 직업은 PRESIDENT 입니다
BLAKE 의 직업은 MANAGER 입니다
CLARK 의 직업은 MANAGER 입니다
JONES 의 직업은 MANAGER 입니다
MARTIN 의 직업은 SALESMAN 입니다
ALLEN 의 직업은 SALESMAN 입니다
TURNER 의 직업은 SALESMAN 입니다
JAMES 의 직업은 CLERK 입니다

Host: 127.0.0.1 Database: orcl Query

```

1  /*
2  SELECT CONCAT (ENAME, SAL)
3  FROM emp;
4  SELECT ENAME || sal
5  FROM emp;
6
7  */
8
9  SELECT CONCAT(ENAME,"의 직업은 ",JOB, "입니다")
10 FROM emp;
11
12
13

```

emp (56r x 1c)

CONCAT(ENAME,"의 직업은 ",JOB, "입니다")
TURNER의 직업은 SALESMAN입니다
JAMES의 직업은 CLERK입니다
WARD의 직업은 SALESMAN입니다
FORD의 직업은 ANALYST입니다
SMITH의 직업은 CLERK입니다
SCOTT의 직업은 ANALYST입니다
ADAMS의 직업은 CLERK입니다
MILLER의 직업은 CLERK입니다
KING의 직업은 PRESIDENT입니다
BLAKE의 직업은 MANAGER입니다

Filter: Regular expression

Quiz 006

사원 테이블에서 직업을 출력하는데 중복된 데이터를 제외하고 출력해 보겠습니다.


JOB
SALESMAN
CLERK
ANALYST
MANAGER
PRESIDENT

```
/*  
SELECT DISTINCT job  
FROM emp;  
*/  
  
SELECT UNIQUE job  
FROM emp;
```

QUIZ 007

ENAME	SAL
SMITH	800
JAMES	950
ADAMS	1100
WARD	1250
MARTIN	1250
MILLER	1300
TURNER	1500
ALLEN	1600
CLARK	2450
BLAKE	2850
JONES	2975
FORD	3000
SCOTT	3000
KING	5000

```
1
2  SELECT UNIQUE ENAME, SAL
3
4  FROM emp
5
6  ORDER BY SAL DESC;
7
```


 emp (14r x 2c)

ENAME	SAL
KING	5,000
FORD	3,000
SCOTT	3,000
JONES	2,975
BLAKE	2,850
CLARK	2,450
ALLEN	1,600
TURNER	1,500
MILLER	1,300
MARTIN	1,250
WARD	1,250
ADAMS	1,100
JAMES	950
SMITH	800

QUIZ 008

ENAME	DEPTNO	SAL
KING	10	5000
CLARK	10	2450
MILLER	10	1300
SCOTT	20	3000
FORD	20	3000
JONES	20	2975
ADAMS	20	1100
SMITH	20	800
BLAKE	30	2850
ALLEN	30	1600
TURNER	30	1500
MARTIN	30	1250
WARD	30	1250
JAMES	30	950

```
8 SELECT ENAME,DEPTNO, SAL
9 FROM emp
10 ORDER BY DEPTNO ASC, SAL DESC;
```

 emp (14r x 3c)

ENAME	DEPTNO	SAL
KING	10	5,000
CLARK	10	2,450
MILLER	10	1,300
SCOTT	20	3,000
FORD	20	3,000
JONES	20	2,975
ADAMS	20	1,100
SMITH	20	800
BLAKE	30	2,850
ALLEN	30	1,600
TURNER	30	1,500
WARD	30	1,250
MARTIN	30	1,250
JAMES	30	950