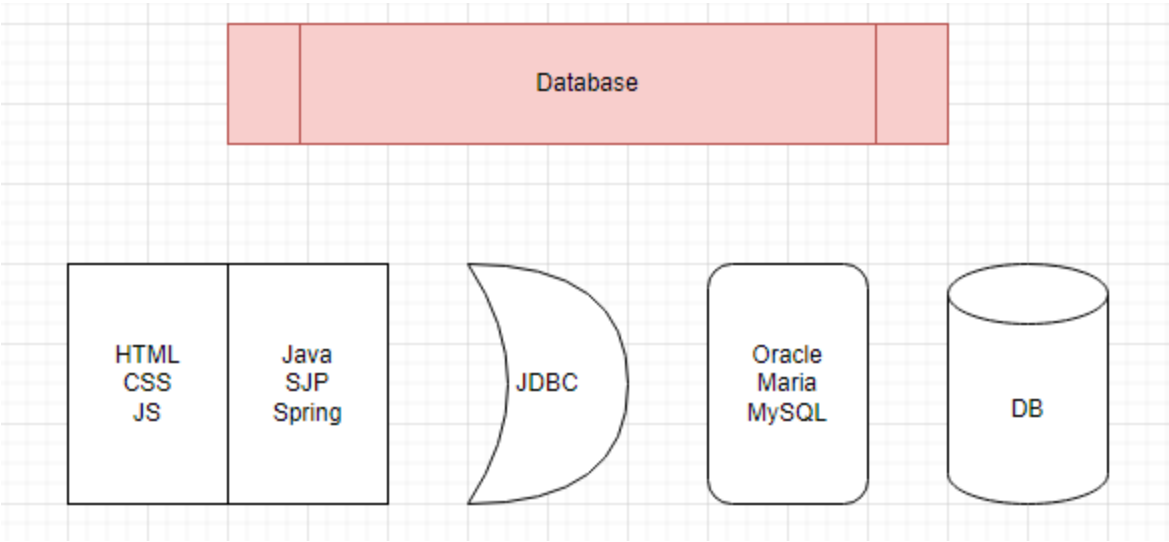
Day 23

작성일시	@2022년 7월 28일 오후 4:43
강의 번호	AUS 101
유형	
자료	
Property	
Date	
복습	☐
속성	

▼ 목록

DBMS(Database Management System) : Oracle, My SQL, MS SQL, Access, Maria DB……

관계형 데이터베이스 Relational DBMS



MariaDB, heidi SQL

SQL(Structured Query Language)관계형 대이터베이스들을 위한 표준 데이터베이스 언어이다.

- DDL(Data Definition Language) : 데이터베이스와 테이블 생성 및 변경 삭제
- DML(Data Management Language(데이터 조작어)) : 데이터 삽입, 검색, 갱신, 삭제
- DCL(Data Control Language(데이터 제어어)) : 데이터베이스 접근 제어 및 사용 권한 관리

▼ 데이터베이스 사용하기

- 전체 데이터 베이스 목록 보기

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

MariaDB [(none)]>
```

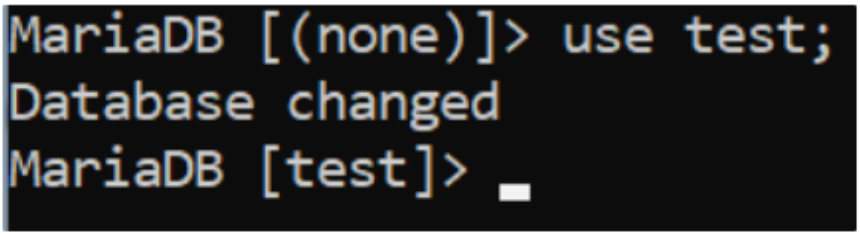
- 데이터베이스 생성하기

```
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
- List



교재 예제설치

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

> https://s3-us-west-2.amazonaws.com/secure.notion-static.com/f0832d5e-73c1-4a31-88f3-cf0adc4cc94d/sakila-db.zip

> https://s3-us-west-2.amazonaws.com/secure.notion-static.com/02c5b262-c29a-4f68-a762-75e539e4b90c/LearningSQLExample.sql

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

자료형

1. 문자 데이터
    a. char 고정길이 : 8글자 데이터를 입력할 때 나머지글자는 모두 공백으로 채워서 저장

b. varchar 가변길이 : 8글자 데이터를 입력할 때 8글자만 저장된다.(주로 사용하는 문자 자료형)

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

2. 텍스트 데이터
   a. 긴 문자열을 저장할 때 사용한다.
      i. text 주로 사용하는 텍스트 자료형
      ii. tinytext

3. 숫자 데이터
   a. int 주로 사용하는 숫자 자료형
   b. tinyint
   c. smallint
   d. bigint

4. 날짜 데이터
   a. timestamp : 현재 날짜와 시간을 자동입력
   b. datetime : 날짜와 시간을 입력받을 때

테이블 작성

1. 설계(Design)

   테이블에 저장할 적절한 항목들과 그 항목들을 저장할 데이터 형과 크기를 설계

이름, 주소, 전화번호, 성별, 음식…

2. 정제(Refinement)

   테이블 작성시에는 기본키 설정이 중요(PRIMARY KEY)

   이름의 경우 성과 이름으로 분리하면 데이터 분석 시 이점을 가질 수 있다.

   주소의 경우에도 하나의 필드로 저장하는 것보다 시, 군, 구 별로 따로 분리하면 데이터 분석 시 이점이 있다.

3. SQL 구문 생성(Building SQL Schema Statements)

```
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),
  state VARCHAR(20),
  country VARCHAR(20),
  postal_code VARCHAR(20),
  CONSTRAINT pk_person PRIMARY KEY (person_id)
 );
```

테이블 수정

1. 데이터 삽입
2.
   a. 데이터를 추가할 테이블 이름
   b. 데이터를 추가할 테이블의 열 이름
   c. 열에 넣을 값

테이블에 데이터 입력

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

테이블 데이터 조회

```
SELECT person_id, fname, lname, eye_color, birth_date FROM person;
```

```
SELECT * FROM person;  - - 필드값들 전체를 조회 (*=전체 필드를 표시)
```

SELECT 테이블 항목 FROM 테이블 명

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

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

SQL 주석 2가지

```
/* 긴주석은
이렇게  표현 */
```

```
- - 한줄주석은 이렇게 표현
```

```
mysql> INSERT INTO favorite_food (person_id, food)
    -> VALUES (1, 'pizza');
Query OK, 1 row affected (0.01 sec)
mysql> INSERT INTO favorite_food (person_id, food)
    -> VALUES (1, 'cookies');
Query OK, 1 row affected (0.00 sec)
mysql> INSERT INTO favorite_food (person_id, food)
    -> VALUES (1, 'nachos');
Query OK, 1 row affected (0.01 sec)
```

```
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');
```

```
mysql> SELECT food
    -> FROM favorite_food
    -> WHERE person_id = 1
    -> ORDER BY food;
```

food 열을 favorite_food라는 테이블로부터 조건은 persin_id=1인 값만 순서를 food열을 오름차순 정렬하여 조회

```
SELECT food
FROM favorite_food
WHERE person_id = 1
ORDER BY food;
```

```
mysql> INSERT INTO person
    ->  (person_id, fname, lname, gender, birth_date,
    ->   street, city, state, country, postal_code)
    -> VALUES (null, 'Susan','Smith', 'F', '1975-11-02',
    ->   '23 Maple St.', 'Arlington', 'VA', 'USA', '20220');
 Query OK, 1 row affected (0.01 sec)
```

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

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

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

```
mysql> DELETE FROM person
    -> WHERE person_id = 2;
```

person 테이블에서 person_id 값이 2인 레코드를 삭제

DELETE FROM person
WHERE person_id = 2;

```
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;

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'))
```

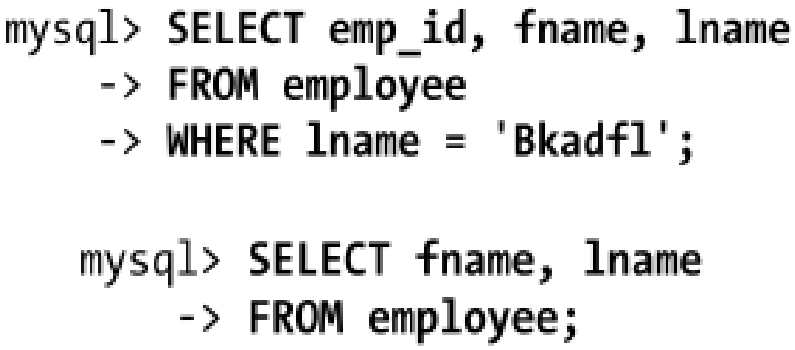
존재하지 않는 외래키 foreign key를 참조할 때 에러 발생

```
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 벗어났을 때 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 - 년 으로 입력되어 에러 발생



쿼리절(Query Clauses)

Clause name	Purpose
Select	Determines which columns to include in the query's result set
From	Identifies the tables from which to draw data and how the tables should be joined
Where	Filters out unwanted data
Group by	Used to group rows together by common column values
Having	Filters out unwanted groups
Order by	Sorts the rows of the final result set by one or more columns

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 - 원한지 않는 그룹을 걸러내는 조건 설정

ORDER BY - 하나 또는 하나 이상의 열들을 기준으로 최종결과의 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 FROM 은 department이라는 하나의 테이블의 모든 열을 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 볼 수도 있다.

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

숫자나 문자를 그냥 출력 ‘ACTIVE’

기존열의 값을 계산한 결과를 출력 emp_id*3.14159

함수를 사용한 결과를 출력 UPPER(lname)

컬럼 별칭

```
mysql> 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 사번,
'ACTIVE' 상태,
UPPER(lname) 대문자성
FROM employee;

내용이 바뀌는 것이 아니라 보여주는 화면만 수정된다.

AS 생략가능!

중복제거 DISTINCT

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

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

상황에 따라 쿼리가 중복된 행을 반환할 수 있다. 고유한 하나의 값만 남기고 나머지는 제거한 값을 확인 할 수 있다.

내용이 바뀌는 것이 아니라 보여주는 화면만 수정된다.

FROM 절

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

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

Permanent Table(영구 테이블) create table로 생성된 테이블

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

Virtual Table(가상 테이블) create view로 생성된 테이블

Temporary Table 파생테이블(←임시테이블)

```
mysql> 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;
```

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;
```

WHERE절

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 |
+--------+---------+---------+------------+-------------+
```

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';
```

GROUP BY절과 HAVING절

```
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 BY열을 기준으로 행들의 값으로 그룹으로 나누고 그 나뉜 그룹에 조건을 적용하는 것이 HAVING 이다.

ORDER BY절

일반적으로 쿼리는 반환된 결과셋의 행은 특정한 순서로 정렬되지 않는다. 결과를 원하는 순서로 정렬하려면 ORDER BY절을 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        |
```

```
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        |
```

정렬의 기준이 여러개인 경우는 첫번째 정렬을 마친 값들 중 동일한 값들만 다시 한 번 정렬

```
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 |
```

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

TEST

```
CREATE TABLE DEPT
(DEPTNO int(10),
DNAME VARCHAR(14),
LOC VARCHAR(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               INT(4) NOT NULL,
ENAME               VARCHAR(10),
JOB                 VARCHAR(9),
MGR                 INT(4) ,
HIREDATE            DATE,
SAL                 INT(7),
COMM                INT(7),
DEPTNO              INT(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);
```

Quiz 001,

```
MariaDB [bank]> SELECT EMPNO, ENAME, SAL
    -> FROM emp;
+-------+--------+------+
| EMPNO | ENAME  | SAL  |
+-------+--------+------+
|  7839 | KING   | 5000 |
|  7698 | BLAKE  | 2850 |
|  7782 | CLARK  | 2450 |
|  7566 | JONES  | 2975 |
|  7654 | MARTIN | 1250 |
|  7499 | ALLEN  | 1600 |
|  7844 | TURNER | 1500 |
|  7900 | JAMES  |  950 |
|  7521 | WARD   | 1250 |
|  7902 | FORD   | 3000 |
|  7369 | SMITH  |  800 |
|  7788 | SCOTT  | 3000 |
|  7876 | ADAMS  | 1100 |
|  7934 | MILLER | 1300 |
+-------+--------+------+
14 rows in set (0.000 sec)
```

SELECT empno, ename, sal

FROM emp;

Quiz 002

사원 테이블을 모든 열(column)들을 전부 출력해 보겠습니다.

```
MariaDB [bank]> SELECT *
    -> FROM emp;
+-------+--------+-----------+------+------------+------+------+--------+
| EMPNO | ENAME  | JOB       | MGR  | HIREDATE   | SAL  | COMM | DEPTNO |
+-------+--------+-----------+------+------------+------+------+--------+
|  7839 | KING   | PRESIDENT | NULL | 1981-11-17 | 5000 | NULL |     10 |
|  7698 | BLAKE  | MANAGER   | 7839 | 1981-05-01 | 2850 | NULL |     30 |
|  7782 | CLARK  | MANAGER   | 7839 | 1981-05-09 | 2450 | NULL |     10 |
|  7566 | JONES  | MANAGER   | 7839 | 1981-04-01 | 2975 | NULL |     20 |
|  7654 | MARTIN | SALESMAN  | 7698 | 1981-09-10 | 1250 | 1400 |     30 |
|  7499 | ALLEN  | SALESMAN  | 7698 | 1981-02-11 | 1600 |  300 |     30 |
|  7844 | TURNER | SALESMAN  | 7698 | 1981-08-21 | 1500 |    0 |     30 |
|  7900 | JAMES  | CLERK     | 7698 | 1981-12-11 |  950 | NULL |     30 |
|  7521 | WARD   | SALESMAN  | 7698 | 1981-02-23 | 1250 |  500 |     30 |
|  7902 | FORD   | ANALYST   | 7566 | 1981-12-11 | 3000 | NULL |     20 |
|  7369 | SMITH  | CLERK     | 7902 | 1980-12-11 |  800 | NULL |     20 |
|  7788 | SCOTT  | ANALYST   | 7566 | 1982-12-22 | 3000 | NULL |     20 |
|  7876 | ADAMS  | CLERK     | 7788 | 1983-01-15 | 1100 | NULL |     20 |
|  7934 | MILLER | CLERK     | 7782 | 1982-01-11 | 1300 | NULL |     10 |
+-------+--------+-----------+------+------------+------+------+--------+
14 rows in set (0.001 sec)
```

SELECT *

FROM emp;

SELECT empno, ename, job, mgr,
hiredate, sal, comm, deptno
FROM emp

Quiz 003

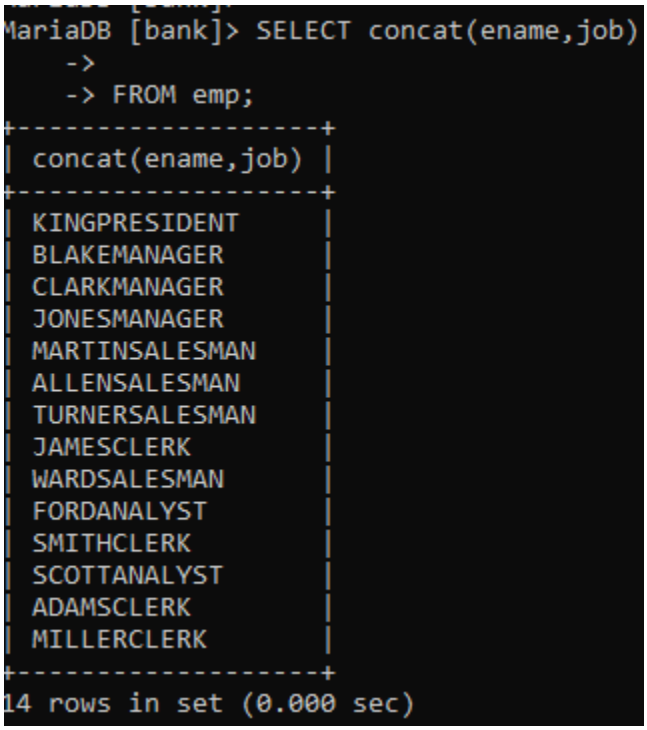
사원 테이블의 사원 번호와 이름과 월급을 출력하는데 컬럼명을 한글로 **'사원 번호'**, **'사원 이름'** 으로 출력해 보겠습니다.

```
MariaDB [bank]> SELECT empno AS 사원번호,
    -> ename AS 사원이름,
    -> sal AS '사원 Salary'
    -> FROM emp;
+--------------+--------------+---------------+
| 사원번호     | 사원이름     | 사원 Salary   |
+--------------+--------------+---------------+
|         7839 | KING         |          5000 |
|         7698 | BLAKE        |          2850 |
|         7782 | CLARK        |          2450 |
|         7566 | JONES        |          2975 |
|         7654 | MARTIN       |          1250 |
|         7499 | ALLEN        |          1600 |
|         7844 | TURNER       |          1500 |
|         7900 | JAMES        |           950 |
|         7521 | WARD         |          1250 |
|         7902 | FORD         |          3000 |
|         7369 | SMITH        |           800 |
|         7788 | SCOTT        |          3000 |
|         7876 | ADAMS        |          1100 |
|         7934 | MILLER       |          1300 |
+--------------+--------------+---------------+
14 rows in set (0.000 sec)
```

띄어쓰기를 사용 할 시 ‘ ‘ 사이에 입력

Quiz 004

사원 테이블의 이름과 월급을 서로 붙여서 출력해 보겠습니다.



```
MariaDB [bank]> SELECT concat(ename,job)
    ->
    -> FROM emp;
+-------------------+
| concat(ename,job) |
+-------------------+
| KINGPRESIDENT     |
| BLAKEMANAGER      |
| CLARKMANAGER      |
| JONESMANAGER      |
| MARTINSALESMAN    |
| ALLENSALESMAN     |
| TURNERSALESMAN    |
| JAMESCLERK        |
| WARDSALESMAN      |
| FORDANALYST       |
| SMITHCLERK        |
| SCOTTANALYST      |
| ADAMSCLERK        |
| MILLERCLERK       |
+-------------------+
14 rows in set (0.000 sec)
```

[오라클에서만 동작]

SELECT ename || sal
FROM emp;

Quiz 004

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

```
MariaDB [bank]> SELECT concat(ename,"의 직업은",job,"입니다.") AS 직업정보
    ->
    -> FROM emp;
+--------------------------------------+
| 직업정보                             |
+--------------------------------------+
| KING의 직업은PRESIDENT입니다.        |
| BLAKE의 직업은MANAGER입니다.         |
| CLARK의 직업은MANAGER입니다.         |
| JONES의 직업은MANAGER입니다.         |
| MARTIN의 직업은SALESMAN입니다.       |
| ALLEN의 직업은SALESMAN입니다.        |
| TURNER의 직업은SALESMAN입니다.       |
| JAMES의 직업은CLERK입니다.           |
| WARD의 직업은SALESMAN입니다.         |
| FORD의 직업은ANALYST입니다.          |
| SMITH의 직업은CLERK입니다.           |
| SCOTT의 직업은ANALYST입니다.         |
| ADAMS의 직업은CLERK입니다.           |
| MILLER의 직업은CLERK입니다.          |
+--------------------------------------+
14 rows in set (0.000 sec)
```

Quiz 005

사원 테이블에서 직업을 출력하는데 중복된 데이터를 제외하고 출력해 보겠습니다.

JOB
SALESMAN
CLERK
ANALYST
MANAGER
PRESIDENT

```
MariaDB [bank]> SELECT UNIQUE job
    -> FROM emp;
+-----------+
| job       |
+-----------+
| PRESIDENT |
| MANAGER   |
| SALESMAN  |
| CLERK     |
| ANALYST   |
+-----------+
5 rows in set (0.000 sec)

MariaDB [bank]>
```

```
MariaDB [bank]> SELECT DISTINCT job
    -> FROM emp;
+-----------+
| job       |
+-----------+
| PRESIDENT |
| MANAGER   |
| SALESMAN  |
| CLERK     |
| ANALYST   |
+-----------+
5 rows in set (0.000 sec)
```

QUIZ 006

이름과 월급을 출력하는데 월급이 낮은 사원부터 출력해 보겠습니다.

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

```
MariaDB [bank]> SELECT ename, sal
    -> FROM emp
    -> ORDER BY sal DESC;
+--------+------+
| ename  | sal  |
+--------+------+
| KING   | 5000 |
| SCOTT  | 3000 |
| FORD   | 3000 |
| JONES  | 2975 |
| BLAKE  | 2850 |
| CLARK  | 2450 |
| ALLEN  | 1600 |
| TURNER | 1500 |
| MILLER | 1300 |
| WARD   | 1250 |
| MARTIN | 1250 |
| ADAMS  | 1100 |
| JAMES  |  950 |
| SMITH  |  800 |
+--------+------+
14 rows in set (0.000 sec)
```

```
MariaDB [bank]> SELECT ename,deptno, sal
    -> FROM emp
    -> ORDER BY deptno ASC, sal DESC;
+--------+--------+------+
| 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 |
| WARD   |     30 | 1250 |
| MARTIN |     30 | 1250 |
| JAMES  |     30 |  950 |
+--------+--------+------+
14 rows in set (0.001 sec)
```