Day23

Date	@01/08/2022
작성일시	@2022년 8월 1일 오전 9:39
강의 번호	BD101
유형	Data
강사명	AustinYoon
강의자료	☐
노션 복습	☐
코딩 복습	☐
주말숙제(교제)	☐
정리	☑

Day 23 Class

Jul

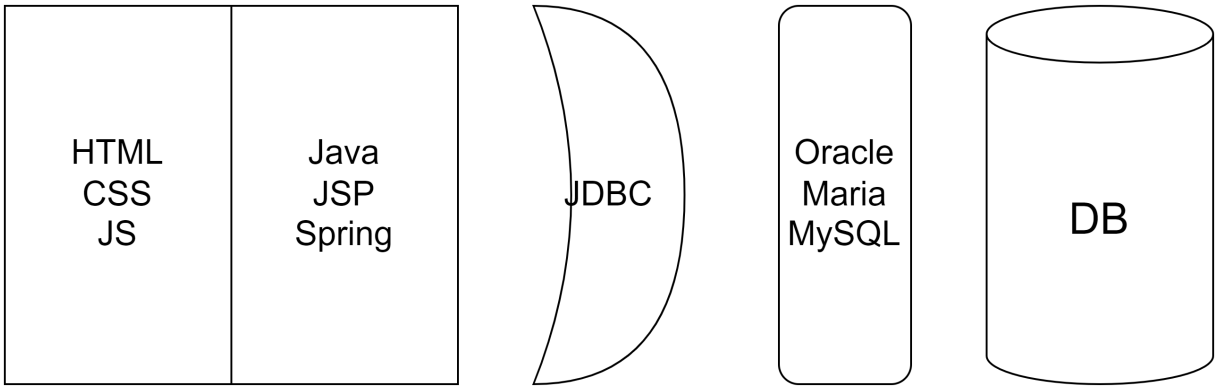
https://s3-us-west-2.amazonaws.com/secure.notion-static.com/c61d2c94-c79a-4b27-8127-d94d10d44474/Learning_SQL_Second_Edition.pdf

DB

DBMS Database Management System : Oracle, My SQL, MS SQL, Access, MariaDB …..

관계형 데이터베이스 Relational DBMS

Database



Maria -

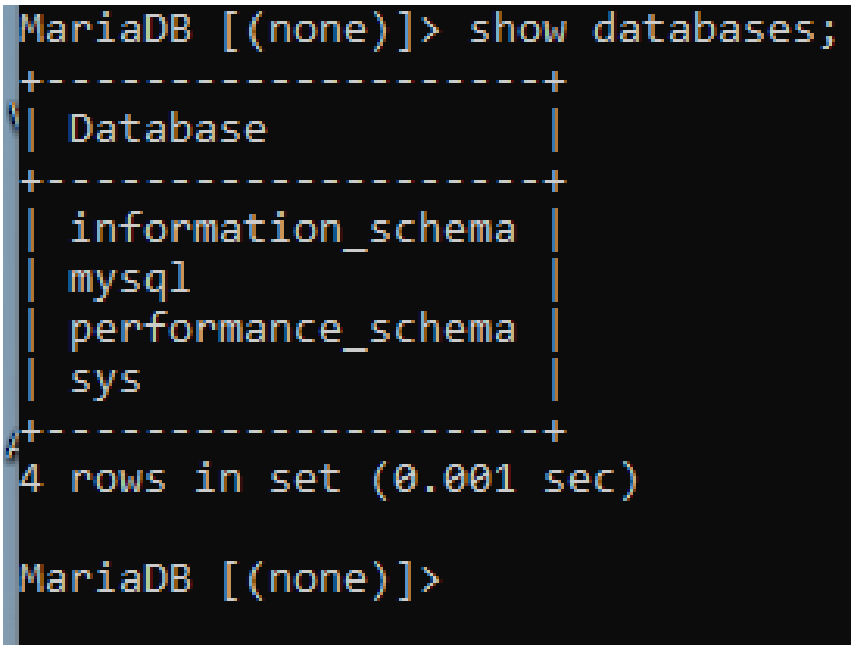
https://mariadb.com/download-confirmation/?group-name=Community Server&release-notes-uri=https%3A%2F%2Fmariadb.com%2Fkb%2Fen%2Fmariadb-1083-release-notes%2F&documentation-uri=https%3A%2F%2Fmariadb.com%2Fkb%2Fen%2Fwhat-is-mariadb-108%2F&download-uri=https%3A%2F%2Fdlm.mariadb.com%2F2314686%2FMariaDB%2Fmariadb-10.8.3%2Fwinx64-packages%2Fmariadb-10.8.3-winx64.msi&product-name=Community Server&download-size=65.25 MB

https://www.heidisql.com/

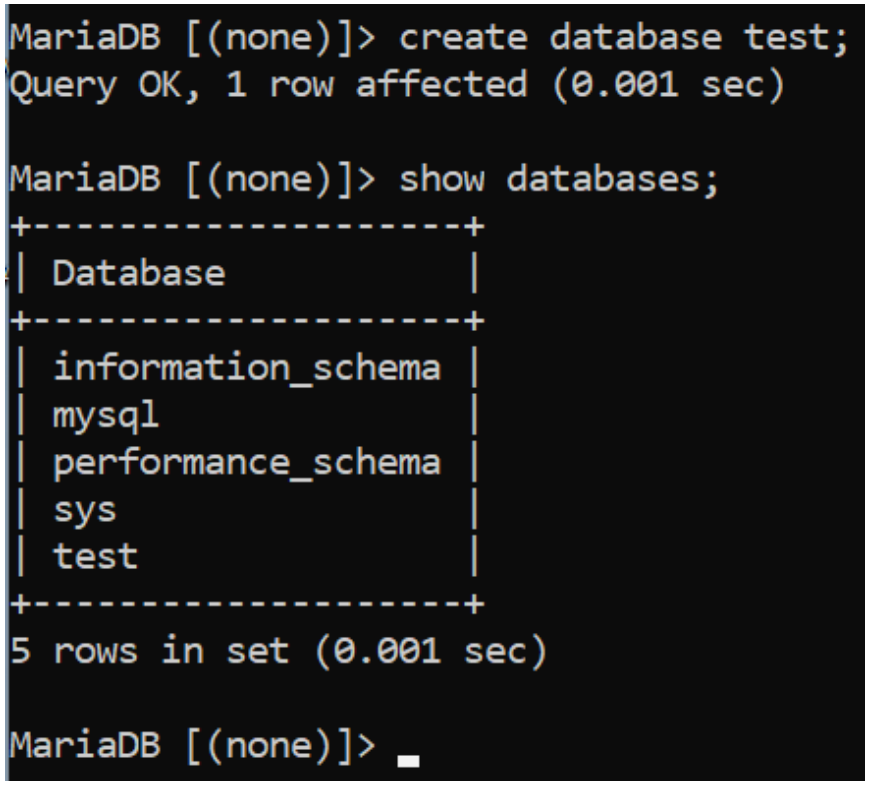
SQL (Structured Query Language) 관계형 데이터베이스들을 위한 표준 데이터베이스 언어이다.

- DDL (Data Definition Language : 데이터 정의어) 데이터베이스와 테이블 생성과 변경 과 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

- 전체 데이터 베이스 목록 보기



- 데이터베이스 생성하기



- 사용할 데이터 베이스 선택

자료형

1. 문자 데이터

```
char 고정길이 : 만약 8글자 데이터를 입력할 때, 나머지 글자는 모두 빈칸으로 채워서 저장
varchar 가변길이 : 8글자 데이터를 입력하면, 8글자만 저장된다. _주로 사용.
```

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

2. text 데이터

   아주 긴 문자열을 저장할 때 사용한다.

   - text 주로 사용하는 텍스트 자료형
   - 아주 작은 글자 사용시 tinytext *MySQL에만 있다.

3. 숫자 데이터

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

4. 날짜 데이터
   - timestamp - 현재 날짜와 시간을 자동입력
   - datetime - 날짜와 시간

태이블 작성

1. 설계(디자인)

   ⇒ 테이블에 저장할 적절한 항목들과 그 항목들을 저장한 데이터 형과 크기를 설계

   이름, 주소, 전화번호, 성, 음식 ….

2. 정제

   ⇒ 이름의 경우에는 성과 이름으로 분리를 하면 데이터 분석 시 이점을 가질 수 있다.
   마찬가지로 주소의 경우에도 하나의 필드로 저정하는 것보다 시,군, 구 별로 따로 분리 하는 것이 좋다.
   ⇒ 데이터를 저장할 때, 기본 키 데이터를 설정하는 것을 추천한다. (학번, 사번, 전화번호 등)_(PRIMARY KEY)

3. SQL구문 생성

테이블 생성

```
CREATE TABLE person
(person_id SMALLINT UNSIGNED,
fname VARCHAR(20),
lname VARCHAR(20),
eye_color ENUM('BR','BL','GR'),
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)
);
```

⇒ 교재 예시 데이터

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/8ce03832-6ee2-4104-8c6d-437b73a4b278/sakila-data.sql

```
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)
);
```

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/078fc056-3734-422b-920c-22a8198d591f/sakila-schema.sql

- 소스 활용

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

테이블 수정

(1) 데이터 삽입

- 데이터를 추가할 테이블 이름
- 데이터를 추가할 테이블의 열이름
- 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');
```

```
-- 전체 데이터 보는 법
SELECT person_id, fname, lname, eye_color, birth_date FROM person;
SELECT * FROM person; -- 두개가 같음
```

```
SELECT person_id, fname, lname, birth_date
FROM person
WHERE person_id = 1;
/* 긴 주석!
-- person 테이블에서 person_id = 1인
-- 조건에 해당하는  person_id, fname, lname, birth_date
-- 필드값들을 조회
*/
```

```
--터너 찾기
SELECT person_id, fname, lname, birth_date
FROM person
WHERE lname = 'Turner';
```

```
--터너가 좋아하는 음식 입력
INSERT INTO favorite_food (person_id, food)
VALUES(1, 'cookies');
INSERT INTO favorite_food (person_id, food)
VALUES(1, 'nachos');
INSERT INTO favorite_food (person_id, food)
VALUES(1, 'pizza');
```

```
--터너가 좋아하는 음식 찾기
SELECT food   --food 열을
FROM favorite_food -- favorite_food 테이블에서
WHERE person_id = 1 -- 조건은 1의 값만
ORDER BY food;  -- 오름차순으로 정리하여 조회
```

```
-- Susan 데이터 추가
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');
```

```
-- 데이터 수정
UPDATE person
  SET street = '1223 Tremont St.',
    city = 'Boston',
    state = 'MA',
    country = 'USA',
    postal_code = '02138'
  WHERE person_id = 1;
```

```
-- 데이터 삭제 'Susan'
DELETE FROM  person   -- person 테이블에서
WHERE person_id = 2;   -- person_id 값이 2인 레코드 삭제
```

```
-- 데이터 제거 (새로고침 눌러서 안되면, 메인 폴더를 선택해서 새로고침, 그래도 남아 있으면 )
DROP TABLE favorite_food;
DROP TABLE person;
```

```
-- 데이터 자세히 보기
DESC customer;
```

에러

- 중복된 고유 키 값을 가진 데이터를 입력하려고 시도할 경우

```
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'
```

- 존재하지 않는 foreign key를 참조할 때의 경우

```
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'))
```

- 열 값 위반, 선언한 데이터형을 벗어나는 경우

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

- 잘못된 날짜 변환_(년-월-일로 입력 x, 월-일-년으로 입력 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
```

⇒bank폴더에서 진행_없으면 생성!

```
--소스코드 활용
use bank;
mysql> SOURCE C:/temp/LearningSQLExample.sql;
```

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/2227cef2-b5ab-465c-8a7a-1d435a97b1c1/LearningSQLExample.sql

C:/temp 위치에 파일 넣기

```
--결과 없는경우
```

```
-결과 있는 경우
```

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

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

- SELECT : 쿼리 결과에 포함 시킬 열을 결정
- FROM : 결과를 검색할 테이블, 테이블들을 접근하는 방법
- WHERE : 원하지 않는 데이터를 걸러내는 조건 설정
- GROUP BY : 공통열 값을 기준으로 행들을 그룹화
- HAVING : 원하지 않는 그룹을 걸러내는 조건 설정
- ORDER BY : 하나또는 하나 이상의 열들을 기준으로 최종 결과의 행들을 정렬

---

1. SELECT

select 절을 완전하게 이해하기 위해서는 from절을 먼저 이해해야한다.

```
SELECT *
FROM department;
/* 이 쿼리에서의 from은 department라는 하나의 테이블의 모든 열을
  결과에 포함해라. */
```

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

```
SELECT dept_id, name
FROM department;
--또는 하나의 열만 선택하여 결과를 볼수 도 있다.
```

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

```
SELECT name
FROM department;
--숫자나 문자를 그냥 출력
```

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

```
SELECT emp_id,
'ACTIVE',
emp_id * 3.14159,
UPPER(lname)
FROM employee;
-- 기존열의 값을 계산한 결과를 출력, 함수를 사용한 결과
```

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

컬럼 별칭

```
--AS를 붙여서 별칭을 만든다. 한글도 잘 됨. AS생략가능 이지만
SELECT emp_id,
'ACTIVE' AS status,
emp_id *3.14149 AS 대문자성
FROM employee;   --자료는 그대로 바꿔서 보여줄뿐
```

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

중복제거

```
SELECT DISTINCT cust_id
FROM account;
-- 상황에 따라 쿼리가 중복된 행을 반환,
-- 고유값 하나의 값만 남기고 나머지는 제거한 값을 보여준다.
```

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

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

---

2. FROM절

- 지금까지 FROM절에 단 하나의 테이블만 지정하였는데 대부분의 실제 SQL구문에서는 하나 이상의 테이블을 목록으로 정의하여 사용된다.
- FROM절은 쿼리에 사용되는 테이블을 명시할 뿐만 아니라 테이블들을 서로 연결하는 수단도 정의하게 된다.

(1) Permanent Table 영구테이블 create table로 생성된 테이블

(2) Temporary Table 임시테이블 서브쿼리로 반환된 행들, 메모리에 임시 저장된 휘발성 테이블

(3) Virtual Table 가상테이블 create view로 생성된 테이블

파생테이블(Temporary Table)

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

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

Virtual Table(View)

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

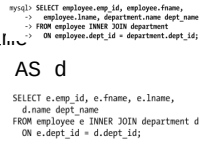
```
mysql> CREATE VIEW employee_vw AS
    -> SELECT emp_id, fname, lname,
    ->   YEAR(start_date) start_year
    -> FROM employee;
```

```
SELECT emp_id, START_year
FROM employee_vw;
```

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

Page 51 테이블 연결

```
SELECT e.emp_id, e.fname,e.lname dept_name
FROM employee AS e INNER JOIN department AS 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.

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

```
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

```
SELECT emp_id, fname, lname, start_date, title
FROM employee
where title = 'Head Teller'
  AND start_date > '2002-01-01';
```

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

Group by절과 Having절

```
SELECT d.name, COUNT(e.emp_id) num_employees
FROM department AS d INNER JOIN employee AS e
  ON d.dept_id = e.dept_id
GROUP BY d.name  --열을 기준으로 행들의 값으로 그룹을 나.
HAVING COUNT(e.emp_id) > 2; --그 나뉜 그룹에 조건을 적용하는 것
```

```
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;
```

ORDER BY절

- 일반적으로 쿼리는 반환된 결과셋의 행은 특정한 순서로 정렬되지는 않는다. 결과를 원하는 순서로 정렬하려면 ORDER BY절을 사용한다.

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

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

```
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, 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 첫 번째 정렬의 마친 값들 중 동일한 값들만 다시 한 번 정렬

```
SELECT open_emp_id, product_cd
FROM account
ORDER BY OPEN_emp_id, product_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        |
```

- 정렬기본은 오름차순으로 적시 하지 않으면 오름차순 정렬이되고, DESC를 적으면 내림차순으로 정렬된다.

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

```
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 |
```

Web상의 SQL

W3Schools Free Online Web Tutorials

With the world's largest web developer site. HTML Tutorial This is a heading This is a paragraph. Try it Yourself W3Schools' famous color picker Play Game Test your skills! Browse our selection of free

https://www.w3schools.com/

문제 풀이 소스코드

```
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);
```

Q1.

사원 테이블에서 사원 번호와 이름과 월급을 출력해 보겠습니다.

```
SELECT empno, ename, sal
  FROM emp;
```

empno	sal	ename
7,839	5,000	KING
7,698	2,850	BLAKE
7,782	2,450	CLARK
7,566	2,975	JONES
7,654	1,250	MARTIN
7,499	1,600	ALLEN
7,844	1,500	TURNER
7,900	950	JAMES
7,521	1,250	WARD
7,902	3,000	FORD
7,369	800	SMITH

Q2.

사원 테이블을 모든 열(column)들을 전부 출력해 보겠습니다.

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

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

emp (14r × 8c)

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

Q3.

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

```
SELECT EMPNO AS 사원번호, ENAME AS 이름,  SAL AS Salary
FROM emp;
```

emp (14r × 3c)

사원번호	이름	Salary
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
7,499	ALLEN	1,600
7,844	TURNER	1,500
7,900	JAMES	950
7,521	WARD	1,250
7,902	FORD	3,000
7,369	SMITH	800

Q4.

사원 테이블의 이름과 월급을 서로 붙여서 출력해 보겠습니다.

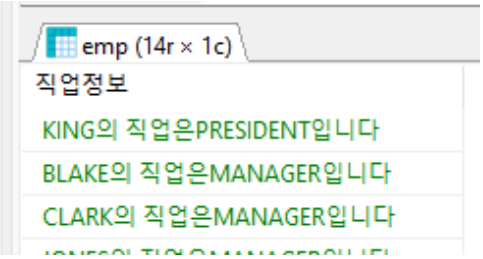
```
SELECT concat( ENAME,  SAL)
FROM emp;

-- SELECT ENAME || SAL *오라클에서만
-- FROM emp;
```

emp (14r × 1c)

concat(ENAME, SAL)
KING5000
BLAKE2850
CLARK2450
JONES2975
MARTIN1250

Q5.원하는 대로 글자와 함께 데이터 출력.

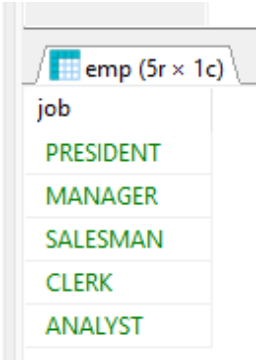
```
SELECT CONCAT( ename,'의 직업은',job,'입니다') AS 직업정보
FROM emp;
```



Q6.

사원 테이블에서 직업을 출력하는데 중복된 데이터를 제외하고 출력해 보겠습니다.

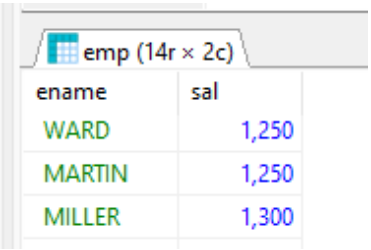
```
/*
SELECT DISTINCT job
FROM emp;
*/
SELECT unique job
  FROM emp;
```



Q7.

이름과 월급을 출력하는데 월급이 낮은 사원부터 출력해 보겠습니다.

```
SELECT ename, sal
  FROM emp
  ORDER BY sal; --오름차순
```



```
SELECT ename, sal
  FROM emp
  ORDER BY sal DESC; --내림차순
```

emp (14r × 2c)

ename	sal
KING	5,000
SCOTT	3,000
FORD	3,000
JONES	2,975
BLAKE	2,850

```
SELECT ename, deptno, sal
  FROM emp
  ORDER BY deptno ASC, sal DESC; --내림차순 안에 오름차순
```

emp (14r × 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