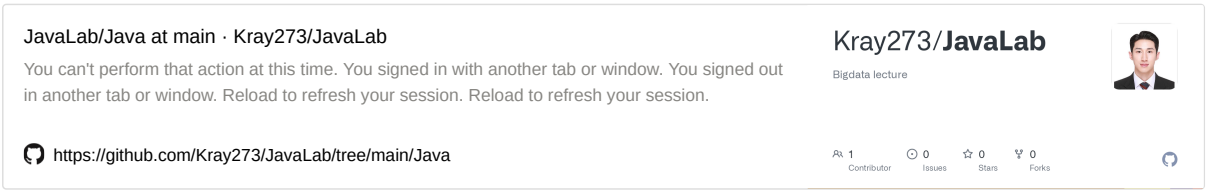
Day(BE)117_DB기본, SELECT,데이터 집계

Date	@2022/12/23
작성일시	@December 23, 2022 10:00 AM
강의 번호	US101
Skilll	Data_SQL Install
Instructor	조성희
Sum up	☑
Review	☐



https://s3-us-west-2.amazonaws.com/secure.notion-static.com/ca0f8eca-fbf2-4b3d-91e1-49a4b5c87a70/day1.pdf

Index

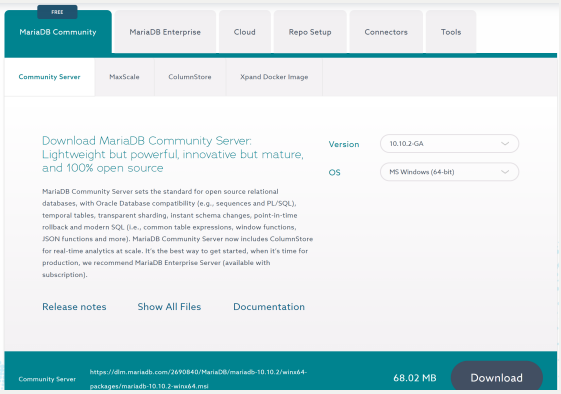
▼ install

마리아 DB 설치

1. 홈페이지 접속

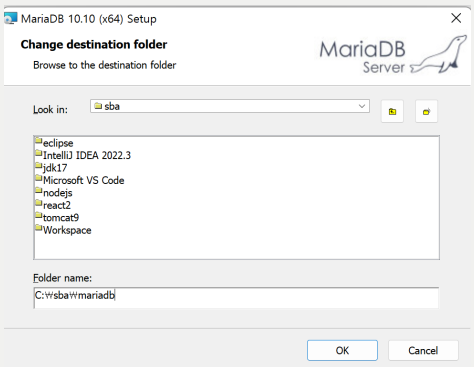
> Open Source Database (RDBMS) for the Enterprise | MariaDB
> We take a different approach to databases for the modern world. Our next generation cloud databases are relational, scale both reads and writes using distributed SQL, support any workload (transactions and analytics), and are
> https://mariadb.com/

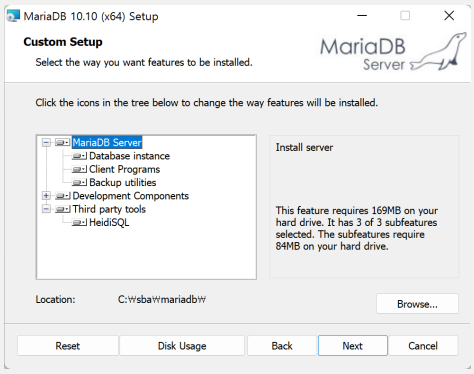
2. 다운로드 _ OS설정 변경



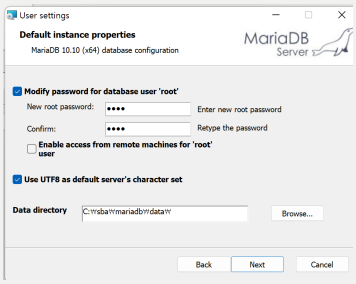
3. SetUp

a. 저장위치 변경

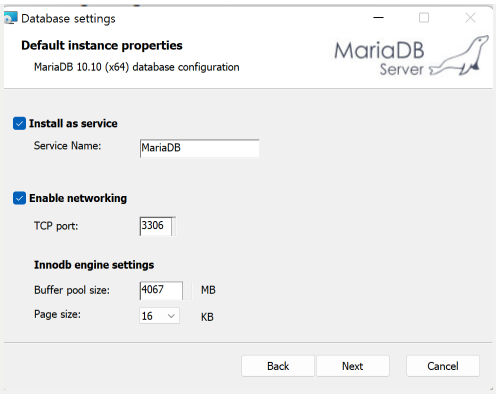




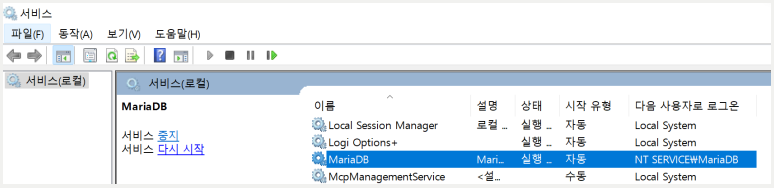
b. 암호 와 UTF-8 클릭 _ 1234



c. 포트번호 확인



d. 설치확인 - 제어판 → 윈도우즈 툴즈 → 서비스

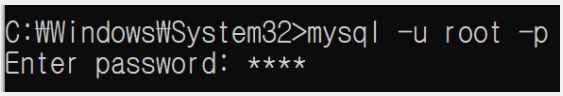


마리아 DB 실행

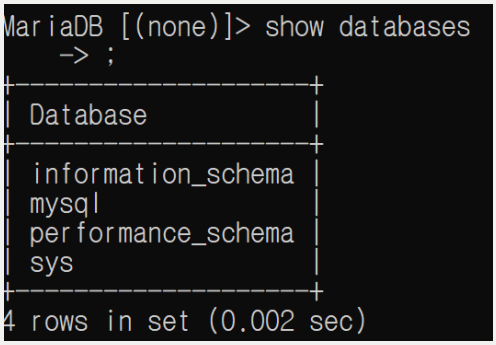
1. 마리아 프롬프트 실행

2. 코드입력

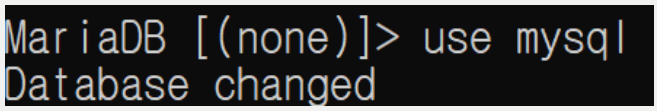
a. 입력(기본계정 root입장) : mysql -u root -p



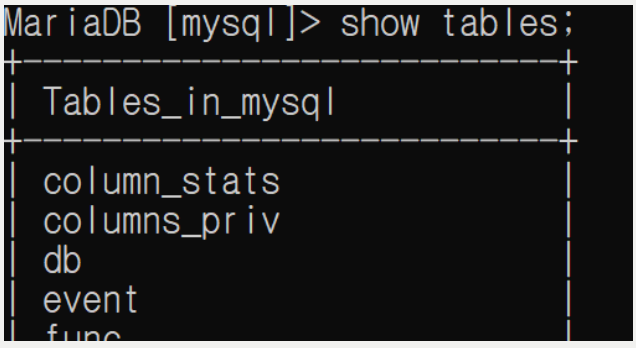
b. 입력(database보기) : show databases;



c. 입력(db사용) : use mysql



d. 입력(테이블 보기) : show tables;



e. 입력(조회) : SELECT user , host FROM user;

```
MariaDB [mysql]> select user , host from user;
+-------------+-----------------+
| User        | Host            |
+-------------+-----------------+
| root        | 127.0.0.1       |
| root        | ::1             |
| root        | desktop-j26er00 |
| mariadb.sys | localhost       |
| root        | localhost       |
+-------------+-----------------+
5 rows in set (0.005 sec)
```

f. 입력(id생성) : create user emp@'%' identified by 'emp';

```
MariaDB [mysql]> create user emp@'%' identified by 'emp';
Query OK, 0 rows affected (0.011 sec)
```

1. _ '%'는 누구든 접근 가능, 대신 localhost 입력시 현재 컴퓨터만 _ identified는 비번설정 ''처리해야함.

g. 입력(모든 권한 부여) : grant all privileges on empdb.* to emp@'%';

```
MariaDB [mysql]> grant all privileges on empdb.* to emp@'%';
Query OK, 0 rows affected (0.011 sec)
```

h. 입력(데이터 베이스 생성) : create database empdb;

```
MariaDB [mysql]> create database empdb;
Query OK, 1 row affected (0.001 sec)
```

i. 입력(계정 로그아웃) : exit

```
MariaDB [mysql]> exit
Bye
```

j. 입력(신규계정 로그인) : mysql -u emp -p

```
C:₩Windows₩System32>mysql -u emp -p
Enter password: ***
Welcome to the MariaDB monitor.  Commands end with ; or ₩g.
Your MariaDB connection id is 4
Server version: 10.10.2-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '₩h' for help. Type '₩c' to clear the current input statement.
```

k. 입력(DB사용) : use empdb;

```
MariaDB [(none)]> use empdb;
Database changed
```

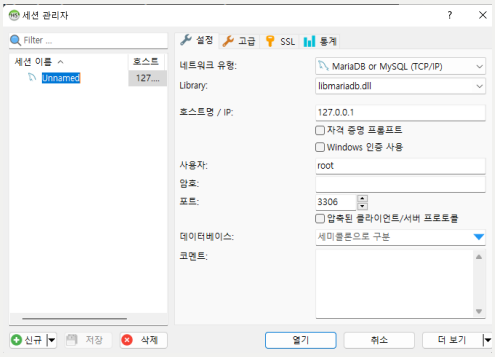
l. 입력(테이블 확인) : show tables;

```
MariaDB [empdb]> show tables;
Empty set (0.001 sec)
```

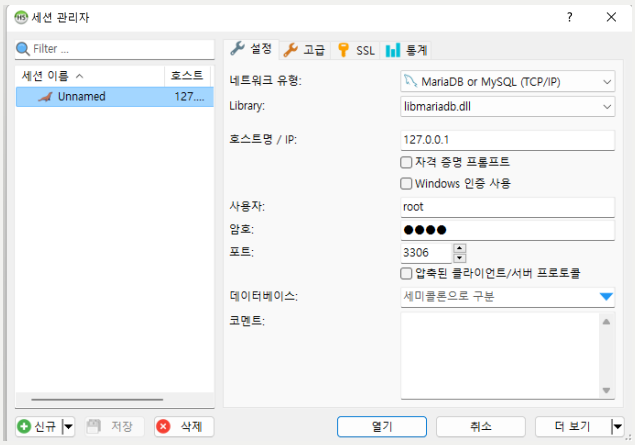
헤이디 사용

1. 헤이디 실행

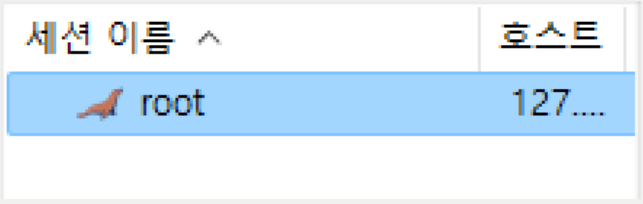
2. 아래 신규 클릭



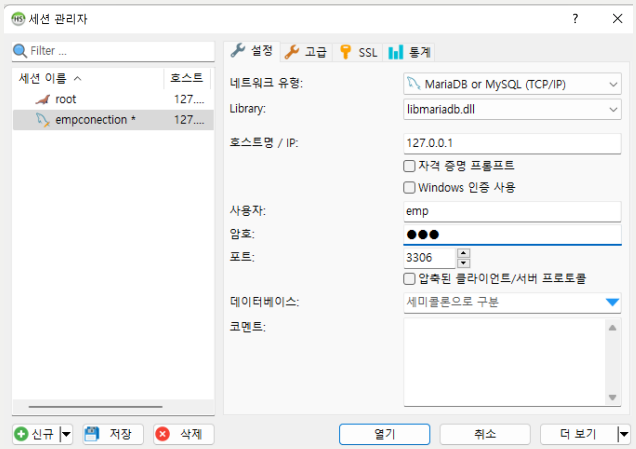
1. 루트 비빌번호 입력하고 열기



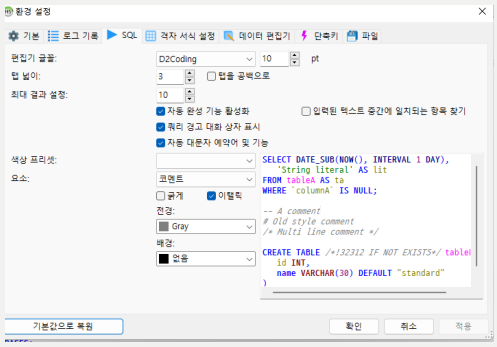
2. 이름변경



3. 동일하게 emp계정 생성



4. 도구 환경설절을 통해 DB 조정;



3. 실습데이터 입력

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/298df351-0966-4790-a6e8-cb41d6951f5b/mysql_hr_insert.sql

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/18874291-e2c9-476a-b2d5-13b9d878f0ff/mysql_hr_create.sql

1. 파일 → sql불러오기 → create파일 클릭 → 인코딩 자동 인식 ok
   a. empdb클릭이 되어있는지 확인!
   b. 코드 실행 안되면 쿼리문창에서 실행
2. 파일 → sql불러오기 → insert파일 클릭 → 인코딩 자동 인식 ok
   a. empdb클릭이 되어있는지 확인!
   b. 코드 실행 안되면 쿼리문창에서 실행

▼ SQL

💡 데이터 ? 의미 잇는 정보 _ 현실

- 데이터 베이스 ⇒ 연관된 데이터 모음
  - 현실의 데이터 ⇒ 마리아DB
- 모델링 : 데이터의 타입 + 길이 + 중복 여부+ null 여부 등을 결정하는 것
- 데이터베이스 표현방법
  - 계층형 구조 : 트리형
  - 네트워크 구조 : 서버
  - 관계형 구조 : RDBMS(오라클, ms sql server, mysql, maria db)
    - 모든 데이터의 관계를 행과 열의 테이블 구조
    - 객체관계형도 있지만 현재 지지부지한 상태… 애매하다.
- RDBMS
  - 사용자가 있어야함 _ 계정 _ emp@'%' / 'emp';
- 스키마 = 데이터베이스 = 연관 테이블의 모음.
- 데이터베이스를 관리할 언어가 SQL(Structured Query Language)

프로그래밍 언어(범용)	자바, java script (jquery, react 포함) 변수 연산자 조건문 반복문 배열
마크업 언어(표시만)	html
데이터베이스 접근 언어(DB)	SQL STRUCTURED ENGLISH QUERY LANGUAGE

- 비용의 문제로 Oracle → MySql → MariaDB
  - MariaDB : 데이터 베이스 설치
    - 접근 SQL
      1. 도스 mysql -u 계정 -p 암호 입력 후 쿼리사용!
      2. heidisql : 바로 쿼리문 사용.
    - 기본 계정 root

mysql -u root -p1234	mysql -u 아이디 -p암호

show databases;
show tables;
select user();
select version();
	use db이름;

다른 사용자 생성-권한	heidisql
create user 아이디@'%' identified by '암호'; grant all ..... on db이름.* to 아이디@'%'; 테이블 생성 – 데이터 저장 수정 삭제 데이터 조회	connection 생성 db리스트 테이블리스트 sql 입력 실행 편집기

▼ SQL문법

문법은 크게 5가지로 나뉜다.

- 사진 캡쳐ddl dml수분

DDL-DATA DEFINITION LANGUAGE 7장	데이터 구조 정의 언어 테이블 생성-CREATE TABLE .... 학생=학번 이름 성적 STU ID NAME SCORE INT CHAR(40) DOUBLE 사용자 생성- CREATE USER .... 데이터베이스 생성-CREATE DATABASE .... ALTER TABLE DROP TABLE
DML- DATA MANIPULATION LANGUAGE 6장	데이터 조작 언어 INSERT / UPDATE / DELETE

DQL- DATA QUERY LANGUAGE 4, 5, 10	데이터 조회 언어 SELECT
DCL – DATA CONTROL LANGUAGE	데이터 제어 언어 테이블 조회 권한 부여 / 회수 테이블 생성 권한 부여 / 회수 단, ROOT 가능 GRANT / REVOKE
TCL – TRANSACTION CONTROL LANGUAGE 19	트랜잭션 제어 언어 COMMIT / ROLLBACK

▼ SELECT

조회

```
SELECT 컬럼명 AS XX, *
FROM 테이블명
[WHERE]
[GROUP BY]
[HAVING]
[ORDER BY];
```

- Select , From절은 필수
- 나머지는 필요에 따라.

```
# 주석
-- 주석
/*
여러줄 주석
*/

-- Employees테이블 조회
SELECT * FROM employees;
```

- 107r * 11c
  - 1r = 1행 = 1개 정보 = 1row = 1tuple
  - 1c = 1열 = 행 표현 정보 묶음 = 1column

SELECT * FROM 테이블명; ---> 테이블의 모든 컬럼, 모든 레코드 조회
SELECT id FROM 테이블명; ---> 테이블의 id 컬럼, 모든 레코드 조회
SELECT id, name FROM 테이블명; ---> 테이블의 id , name 컬럼, 모든 레코드 조회

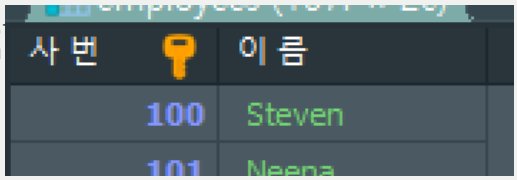
```
SELECT employee_id, first_name FROM employees;
```

```
SELECT employee_id AS 사번, first_name AS 이름 FROM employees;
```

SELECT id as 사번, name as 이름 FROM 테이블명; ---> 테이블의 id , name 컬럼, 모든 레코드 조회

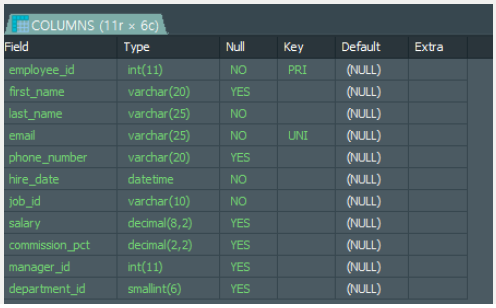
alias-별칭

```
SELECT employee_id AS 사 번, first_name AS 이 름 FROM emp
SELECT employee_id AS '사 번', first_name AS '이 름' FROM
```

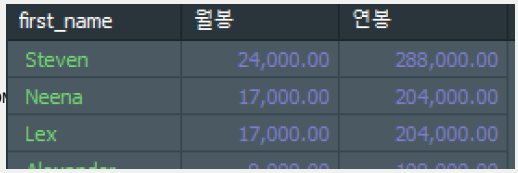


```
DESCRIBE employees;
DESC employees;
```

- employees 테이블의 구성요소 등을 볼수 있는 SQL : DESCRIBE
- DESCRIBE = DESC



```
--(8,2) DECIMAL은 실수(자리, 소수점 자리)
SELECT first_name, salary FROM employees;
-- 연봉 , salary*12
SELECT first_name,salary AS 월봉, salary *12 AS 연봉  FROM
```

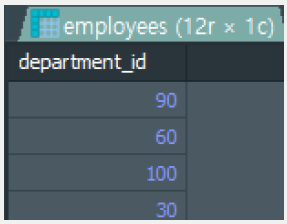


first_name	월봉	연봉
Steven	24,000.00	288,000.00
Neena	17,000.00	204,000.00
Lex	17,000.00	204,000.00

- 숫자데이터는 연산 가능

```
--  사원이 속한 부서코드 종류 조회. 즉 중복제거
SELECT DISTINCT department_id FROM employees;
```

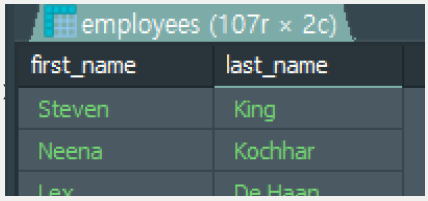
- DISTINCT 는 중복제거



employees (12r × 1c)

department_id
90
60
100
30

```
SELECT first_name, last_name FROM employees;
-- 성은 xxx이고 이름은 yyy입니다.
SELECT CONCAT("성은", first_name,"이고 이름은 ",last_name, "입니다.")
```



employees (107r × 2c)

first_name	last_name
Steven	King
Neena	Kochhar
Lex	De Haan

- 마리아는 결합 기능이 없은 따라서 Concat이라는 함수 사용

```
자바 -> '+'
오라클 ==> '||'
select "성은 " || first_name || " 이고 이름은 " || last_name || " 입니다"
from employees;
```

CONCAT("성은", first_name,"이고 이름은 ",last_name, "...

성은Steven이고 이름은 King입니다.
성은Neena이고 이름은 Kochhar입니다.
성은Lex이고 이름은 De Haan입니다.

where 컬럼명 연산자 값

연산자

산술연산자	+ - * / (숫자타입)
비교연산자	=, !=(<>), >, >=, <, <=
논리연산자	and or not
유사패턴처리연산자	like % , _(문자타입)
목록연산자	in(,,,,,)--> "=" or
범위연산자	between ? and ??
null연산자	is null is not null

select 3.컬럼추출
from 1.테이블명
where 2.조건만족레코드추출

```
-- 레코드 갯수 제한(부서원 번호 80, 부서원 성)
SELECT department_id, last_name FROM employees
WHERE department_id = 80;
```

employees (34r × 2c)

department_id	last_name
80	Russell
80	Partners
80	Errazuriz

- 사원 테이블에서 급여 10000 이상이고 사번 200 미만 사원의 사번, 급여, 이름 조회

```
SELECT employee_id, salary, first_name FROM employees
WHERE salary >= 10000 AND employee_id < 200 ;
```

employees (16r × 3c)

employee_id	salary	first_name
100	24,000.00	Steven
101	17,000.00	Neena

- 사원 테이블에서 급여 10000 ~ 15000 사이 사원의 급여, 이름 조회

```
-- 사원 중 급여가 10000~15000사이 사원의 급여, 이름 조회
SELECT salary, first_name FROM  employees
WHERE salary >= 10000 AND salary <= 10000;

SELECT salary, first_name FROM  employees
WHERE salary BETWEEN 10000 AND 15000;
```

salary	first_name
12,008.00	Nancy
11,000.00	Den
14,000.00	John

- 동일 결과

- 사번 10, 30, 200, 250, 300, 400 인 사원의 급여 10% 인상 조회

```
-- 사번 100,200,150, 222인 사원의 급여  10% 인상 조회
SELECT employee_id, salary * 1.1 AS "인상 급여" FROM emplo
WHERE employee_id IN(100,200,150, 222);
```

- IN 연산자는 or와 동일
  - 222번 없음으로 출력 제외

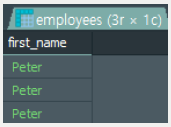
employees (3r × 2c)

employee_id	인상 급여
100	26,400.000
150	11,000.000
200	4,840.000

-필수타입

정수	int
실수	decimal
문자열	char / varchar '' ""
날짜	date '' ""

```
-- db(maria db는  대소문자 구분 없다. ''ㅡ"" 상관없다.)
SELECT first_name, FROM employees
WHERE first_name ='steven';
-- s로 시작하는 사람
SELECT first_name FROM employees
WHERE first_name LIKE 's%';
-- er로 끝나는 사람
SELECT first_name FROM employees
WHERE first_name LIKE '%er';
-- 이름이 5글자 이면서 er 끝나는 사람
SELECT first_name FROM employees
WHERE first_name LIKE '___er';
```

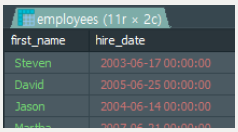


```
SELECT first_name FROM employees
WHERE first_name LIKE "%er" AND CHAR_LENGTH(first_name) >= 5 AND CHAR_LENGTH(first_name) <= 7;
```

```
-- 입사일 조회 datetime
SELECT first_name, hire_date FROM employees;
-- 2002년 입사자의 이름, 입사일 조회
SELECT first_name, hire_date FROM employees
WHERE hire_date LIKE '2002%';
-- 6월  입사자의 이름, 입사일 조회
SELECT first_name, hire_date FROM employees
WHERE hire_date LIKE '_____06%';
```

first_name	hire_date
Steven	2003-06-17 00:00:00
Neena	2005-09-21 00:00:00
Lex	2001-01-13 00:00:00
Alexander	2006-01-03 00:00:00

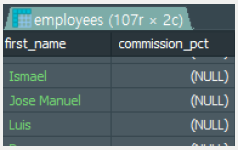
first_name	hire_date
Nancy	2002-08-17 00:00:00
Daniel	2002-08-16 00:00:00
Den	2002-12-07 00:00:00
Susan	2002-06-07 00:00:00



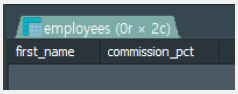
first_name	hire_date
Steven	2003-06-17 00:00:00
David	2005-06-25 00:00:00
Jason	2004-06-14 00:00:00

- _ 는 한문자
- % 는 뭐든!

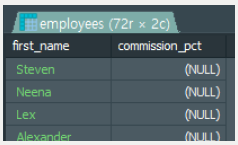
```
-- null인 사람만 조회
SELECT first_name, commission_pct FROM employees;

SELECT first_name, commission_pct FROM employees
WHERE commission_pct = NULL; -- 아무것도 조회 안됨.

SELECT first_name, commission_pct FROM employees
WHERE commission_pct IS NULL;

-- null이 아닌 사람만 조회
SELECT first_name, commission_pct FROM employees
WHERE commission_pct IS NOT NULL;
```



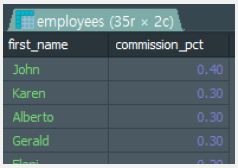
first_name	commission_pct
Ismael	(NULL)
Jose Manuel	(NULL)
Luis	(NULL)



first_name	commission_pct



first_name	commission_pct
Steven	(NULL)
Neena	(NULL)
Lex	(NULL)



first_name	commission_pct
John	0.40
Karen	0.30
Alberto	0.30
Gerald	0.30

- (null) 은 키워드

```
SELECT employee_id FROM employees
ORDER BY employee_id ASC LIMIT 3 , 7; -- 3번부터 7개만
```

employees (7r × 1c)

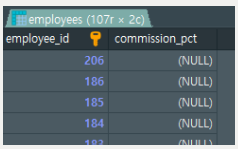
employee_id
103
104
105
106
107

- 오라클에서 동일한 기능 수행시_ 서브쿼리 사용

```
select * ↵
from(select rownum r ↵
from (select employee_id from employees↵
order by hire_date↵
))↵
r >=5 and r <=10↵
```

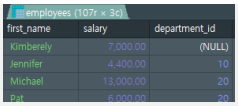
```
-- 정렬
-- null 나중
SELECT employee_id,commission_pct FROM employees
ORDER BY commission_pct DESC;
-- null 우선
SELECT employee_id,commission_pct FROM employees
ORDER BY commission_pct;
-- null 우선 역순(오라클만 가능)
```

employee_id	commission_pct
145	0.40
158	0.35
157	0.35



employee_id	commission_pct
206	(NULL)
186	(NULL)
185	(NULL)
184	(NULL)

```
SELECT employee_id,commission_pct FROM employees
ORDER BY commission_pct DESC nulls FIRST;

-- 부서코드 오름차순, 동일 부서코드인 경우 급여 많은 사원부터 조회
SELECT first_name, salary, department_id FROM employees
ORDER BY department_id, salary DESC;
```

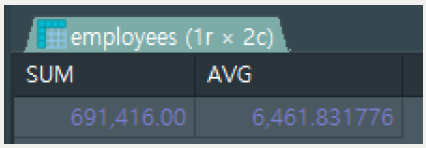


▼ 데이터집계

함수(집계함수 = 그룹함수)_ 1개의 결과만 리턴한다!

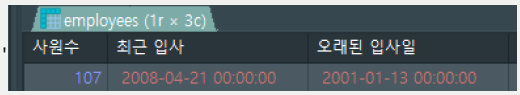
-함수(집계함수 = 그룹함수)	
count	조회 레코드 갯수 리턴
sum	조회 레코드 총합 리턴(숫자컬럼)
avg	조회 레코드 평균 리턴(숫자컬럼)
max	조회 레코드 최대값(숫자큰,문자뒤,날짜최근)
min	조회 레코드 최대값(숫자작,문자앞,날짜최근더멀다)

```
-- 급여의 총합 평균
SELECT SUM(salary) 'SUM', AVG(salary) 'AVG' FROM employees;
```

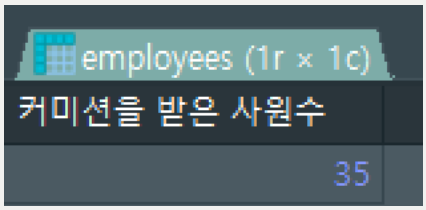



```
SELECT SUM(salary) 'SUM', AVG(salary) 'AVG', COUNT(salar
    MAX(salary)'최대 월급', MIN(salary)'최저 월급'
FROM employees;
```


```
-- 날짜 타입
SELECT COUNT(hire_date) '사원수', MAX(hire_date)'최근 입사'
FROM employees;
```

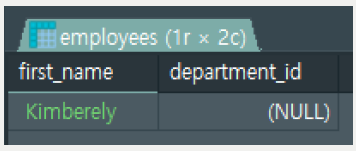



```
-- 커미션(null이 다수 존재 )
SELECT COUNT(commission_pct) ' 커미션을 받은 사원수'
FROM employees; -- null은 카운트에서 제외
```




```
SELECT COUNT(*) '전체 레코드 수'
FROM employees;
```


```
SELECT first_name, department_id FROM employees
WHERE department_id IS null;
```



```
SELECT COUNT(department_id) '소속부서가 있는 사람 수'
FROM employees; -- null을 제외
```

employees (1r × 1c)

소속부서가 있는 사람 수
106

```
-- 사원이름 , 전체 사원급여 총합 조회
SELECT first_name, SUM(salary) FROM employees; -- 의도한
-- 집계함수 외 select 다른 컬럼 조회 불가
```

employees (1r × 2c)

first_name	SUM(salary)
Steven	691,416.00

```
-- 부서코드 종류의 갯수, null제외 됨.
SELECT count(DISTINCT department_id) FROM employees;
```

employees (1r × 1c)

count(DISTINCT department_id)
11

```
-- 50번 부서의 사원 급여의 총합 조회
SELECT SUM(salary) FROM employees
WHERE department_id = 50;
```

employees (1r × 1c)

SUM(salary)
156,400.00

```
-- 각 부서별 사원 급여의 총합 조회
-- 집계함수 외 select 다른 컬럼 조회 불가
-- 단 group by 칼럼은 제외
SELECT DISTINCT department_id, SUM(salary) FROM employees
GROUP BY department_id;
```

employees (12r × 2c)

department_id	SUM(salary)
(NULL)	7,000.00
10	4,400.00
20	19,000.00

```
-- 각 부서별 사원 급여의 총합 조회 단, 부서코드 없는 사람 제외
SELECT DISTINCT department_id, SUM(salary) FROM employee
GROUP BY department_id
HAVING department_id IS NOT NULL;
-- having은 group by 절의 조건으로 group by 뒤에 위치해야함.

SELECT DISTINCT department_id, SUM(salary) FROM employees
WHERE department_id IS NOT NULL
GROUP BY department_id; -- 위와 동일한 결과로 where은 group by 앞에
```

employees (11r × 2c)

department_id	SUM(salary)
10	4,400.00
20	19,000.00
30	24,900.00

employees (11r × 2c)

department_id	SUM(salary)
10	4,400.00
20	19,000.00
30	24,900.00

```
-- 각 부서별, 직종별 부서 사원 급여 조회 단, 부서코드 없는 사원 제외
SELECT DISTINCT department_id, job_id, SUM(salary) FROM
WHERE department_id IS NOT NULL AND job_id IS NOT NULL
GROUP BY department_id, job_id;
```

employees (19r × 3c)

department_id	job_id	SUM(salary)
10	AD_ASST	4,400.00
20	MK_MAN	13,000.00
20	MK_REP	6,000.00
30	PU_CLERK	13,900.00

```
SELECT 조회컬럼명/ 집계함수
FROM 테이블명
WHERE 조회조건식 레코드 추출
GROUP BY 집계함수 그룹컬럼명 , ,
HAVING 집계함수 조건식
ORDER BY 정렬컬럼명 [asc|desc] , , , ;
```

```
-- 각 부서별 부서 사원 급여 총합조회
-- 단 부서코드 없는 사원 제외, 급여총합 10만 이상인 부서만 조회.
SELECT DISTINCT department_id, SUM(salary) FROM employee
WHERE department_id IS NOT NULL
GROUP BY department_id
HAVING SUM(salary) >= 10000;
```

employees (2r × 2c)

department_id	SUM(salary)
50	156,400.00
80	304,500.00

```
SELECT DISTINCT department_id, SUM(salary) FROM employee
WHERE department_id IS NOT NULL
GROUP BY department_id
HAVING SUM(salary) >= 100000
ORDER BY SUM(salary) DESC;  -- order by는 마지막에 작성
```

employees (2r × 2c)

department_id	SUM(salary)
80	304,500.00
50	156,400.00

연습문제 풀이

▼ 기본

-기본--

1. 직원 중에서 연봉이 170000 이상인 직원들의 이름, 연봉을 조회하시오. 연봉은 급여(salary)에 12를 곱한 값입니다.
단, 이름은 "이름", 연봉은 "월급의 12배"로 출력되도록 조회하시오.

```
SELECT First_name AS '이름', salary * 12 '연봉' FROM employees
WHERE salary * 12 >= 170000;
```

employees (3r × 2c)

이름	연봉
Steven	288,000.00
Neena	204,000.00
Lex	204,000.00

2. 직원 중에서 manager_id가 없는 직원의 이름과 급여를 조회하시오.

```
SELECT First_name '이름', salary '급여' , manager_id FROM
WHERE manager_id IS NULL;
```

employees (1r × 3c)

이름	급여	manager_id
Steven	24,000.00	(NULL)

3. 2004년 이전에 입사한 직원의 이름, 급여, 입사일을 조회하시오.

```
SELECT First_name '이름', salary '급여' , hire_date '입사일
WHERE hire_date < '2005-01-01 00:00:00'
ORDER BY hire_date DESC;
```

employees (14r × 3c)

이름	급여	입사일
Trenna	3,500.00	2003-10-17 00:00:00
Jennifer	4,400.00	2003-09-17 00:00:00

4. departments 테이블에서 부서코드, 부서명을 조회하시오.

```
DESC departments;
SELECT department_id '부서코드', department_name '부서명' F
```

departments (27r × 2c)

부서코드	부서명
230	IT Helpdesk

5. jobs 테이블에서 직종코드와 직종명을 조회하시오.

```
DESC jobs;
SELECT job_id '직종코드', job_title '직종명' FROM jobs;
```

jobs (19r × 2c)

직종코드	직종명
AC_ACCOUNT	Public Accountant
AC_MGR	Accounting Manager
AD_ASST	Administration Assistant

▼ 논리연산자

- 논리연산자 --

1. 80, 50 번 부서에 속해있으면서 급여가 13000 이상인 직원의 이름, 급여, 부서id를 조회하시오.

```
DESC employees;
SELECT First_name '이름', salary '급여' , department_id '
WHERE department_id IN (80, 50) AND salary >= 13000;
```

employees (2r × 3c)

이름	급여	부서 ID
John	14,000.00	80
Karen	13,500.00	80

2. 2005년 이후에 입사한 직원들 중에서 급여가 1300 이상 20000 이하인 직원들의 이름, 급여, 부서id, 입사일을 조회하시오.

```
SELECT First_name '이름', salary '급여' ,department_id '부
FROM employees
WHERE hire_date >= '2005-01-01 00:00:00' AND salary BETW
ORDER BY salary DESC;
```

employees (83r × 4c)

이름	급여	부서 ID	입사일
Neena	17,000.00	90	2005-09-21 00:00:00
Karen	13,500.00	80	2005-01-05 00:00:00
Alberto	12,000.00	80	2005-03-10 00:00:00

▼ 비교 연산자

- SQL 비교연산자 --

1. 80, 50 번 부서에 속해있으면서 급여가 13000 이상인 직원의 이름, 급여, 부서id를 조회하시오.

```
DESC employees;
SELECT First_name '이름', salary '급여' , department_id '부서
WHERE department_id IN (80, 50) AND salary >= 13000;
```

employees (2r × 3c)

이름	급여	부서 ID
John	14,000.00	8(
Karen	13,500.00	8(

2. 2005년 이후에 입사한 직원들 중에서 급여가 1300 이상 20000 이하인 직원들의 이름, 급여, 부서id, 입사일을 조회하시오.

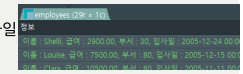
```
SELECT First_name '이름', salary '급여' ,department_id '부서 I
FROM employees
WHERE hire_date >= '2005-01-01 00:00:00' AND salary BETWEEN
ORDER BY salary DESC;
```

employees (83r × 4c)

이름	급여	부서 ID	입사일
Neena	17,000.00	90	2005-09-21 00:00:(
Karen	13,500.00	80	2005-01-05 00:00:(
Alberto	12,000.00	80	2005-03-10 00:00:(

3. 2005년도 입사한 직원의 정보(이름, 급여, 부서코드, 입사일)만 출력하시오.

```
SELECT CONCAT("이름 : ", First_name,", 급여 : ", salary,", 부서 : ", department_id,", 입사일
FROM employees
WHERE hire_date LIKE '2005%'
ORDER BY hire_date DESC;
```



4. 직종이 clerk 군인 직원의 이름, 급여, 직종코드를 조회하시오. (clerk 직종은 job_id에 CLERK을 포함하거나 CLERK으로 끝난다.)

```
SELECT First_name '이름', salary '급여' ,job_id '직종 ID'
FROM employees
WHERE job_id like '%clerk%';
```

employees (45r × 3c)

이름	급여	직종 ID
Alexander	3,100.00	PU_CLERK
Shelli	2,900.00	PU_CLERK
Sigal	2,800.00	PU_CLERK

5. 12월에 입사한 직원의 이름, 급여, 입사일을 조회하시오.

```
SELECT First_name '이름', salary '급여' , hire_date '입사일 '
WHERE hire_date like '_____12%'
ORDER BY hire_date DESC;
```

employees (7r × 3c)

이름	급여	입사일
Randall	2,500.00	2007-12-19 00:00:0(
Ki	2,400.00	2007-12-12 00:00:0(
Luis	6,900.00	2007-12-07 00:00:0(

6. 이름에 le 가 들어간 직원의 이름, 급여, 입사일을 조회하시오.

```
SELECT First_name '이름', salary '급여' , hire_date '입사일 '
WHERE First_name LIKE '%le%'
ORDER BY hire_date DESC;
```

employees (10r × 3c)

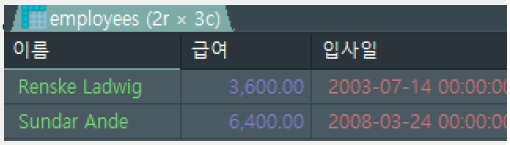
이름	급여	입사일
Eleni	10,500.00	2008-01-29 00:00:00
Charles	6,200.00	2008-01-04 00:00:00

7. 이름이 m으로 끝나는 직원의 이름, 급여, 입사일을 조회하시오.

```
SELECT First_name '이름', salary '급여' , hire_date '입사일 ' FROM emplo
WHERE First_name LIKE '%m'
ORDER BY hire_date DESC;
```

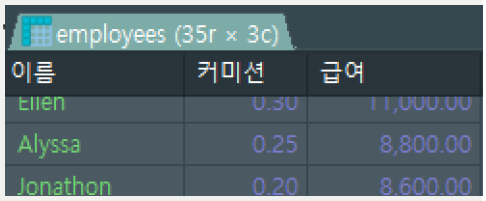
8. 이름의 세번째 글자가 d인 이름, 급여, 입사일을 조회하시오.

```
SELECT concat(First_name,' ',last_name) '이름', salary '급여'
WHERE last_name LIKE '__d%';
```



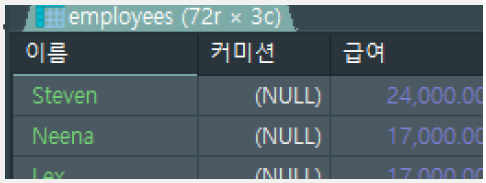
9. 커미션을 받는 직원의 이름, 커미션, 급여를 조회하시오.

```
SELECT First_name '이름', commission_pct '커미션' , salary '급여'
FROM employees
WHERE commission_pct IS NOT NULL;
```



10. 커미션을 받지 않는 직원의 이름, 커미션, 급여를 조회하시오.

```
SELECT First_name '이름', commission_pct '커미션' , salary '급여'
WHERE commission_pct IS NULL;
```



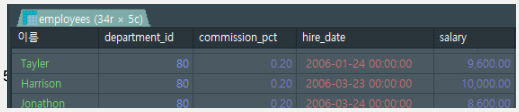
▼ 기타

-기타

1. 30, 50, 80 번 부서에 속해있으면서, 급여를 5000 이상 17000 이하를 받는 직원을 조회하시오.

   단, 커미션을 받지 않는 직원들은 검색 대상에서 제외시키며, 먼저 입사한 직원이 먼저 출력되어야 하며 입사일이 같은 경우 급여가 많은 직원이 먼저 출력되도록 하시오.

```
SELECT First_name '이름', department_id, commission_pct,
FROM employees
WHERE department_id IN (30, 50, 80) AND salary BETWEEN 5
  AND commission_pct IS NOT NULL
ORDER BY hire_date, salary DESC;
```



2. 각 부서별 최대급여와 최소급여를 조회하시오.

   단, 최대급여와 최소급여가 같은 부서는 직원이 한명일 가능성이 높기때문에 조회결과에서 제외시킨다.

```
SELECT department_id '부서ID', MAX(salary)'최대급여' , MIN(sala
FROM employees
GROUP BY department_id
HAVING MAX(salary) != MIN(salary);
```