

1. SELECT 구문을 사용한 데이터 검색

1. 테이블 구조 조회

```
DESCRIBE employees
```

2. 테이블로부터 데이터 검색

1) select 구문

```
SELECT *  
FROM departments;
```

```
SELECT department_id, location_id  
FROM departments;
```

```
SELECT last_name, salary, 12*salary+100  
FROM employees;
```

1

```
SELECT last_name, salary, 12*(salary+100)  
FROM employees;
```

2

2) null값이란?

- [참고] commission_pct 값 update 후 작업하기!
- 사용할 수 없는 값, 알려지지 않은 값, 할당받지 못한 값, 모르는 값, 아직 정의되지 않은 값 등...
- null은 0(zero)나 공백과는 다른 특수한 값, 모든 데이터타입에 사용 가능함.

```
SELECT last_name, job_id, salary, commission_pct  
FROM employees;
```

- 산술식에 null값이 포함되어 있는 경우 산술식 결과도 null이 출력됨.

```
SELECT last_name, 12*salary*commission_pct  
FROM employees;
```

3) Column Alias

① 컬럼명 AS alias

② 컬럼명 alias

③ 컬럼명 [AS] "Alias" => 대소문자 구분, 공백 포함, 특수문자 포함을 원하는 경우

MySQL

```
SELECT last_name AS name, commission_pct comm
FROM employees;
```

	NAME	COMM
1	Whalen	(null)
2	Hartstein	(null)
3	Fay	(null)

...

```
SELECT last_name "Name", salary*12 "Annual Salary"
FROM employees;
```

	Name	Annual Salary
1	Whalen	52800
2	Hartstein	156000
3	Fay	72000

4) DISTINCT 키워드 : 중복된 값을 자동으로 제거해 주는 키워드

```
SELECT DISTINCT department_id
FROM employees;
```

<연습문제>

1. employees 테이블로부터 employee_id, last_name, job_id, hire_date를 출력하되 컬럼 제목을 각각 Emp #, Employee, Job, Hire Date로 지정하여 출력하시오.

	Emp #	Employee	Job	Hire Date
1	200	Whalen	AD_ASST	17-SEP-87
2	201	Hartstein	MK_MAN	17-FEB-96
3	202	Fay	MK_REP	17-AUG-97
4	205	Higgins	AC_MGR	07-JUN-94
5	206	Gietz	AC_ACCOUNT	07-JUN-94

...

19	176	Taylor	SA_REP	24-MAR-98
20	178	Grant	SA_REP	24-MAY-99

2. employees 테이블로부터 사원들이 담당하고 있는 업무 리스트를 출력하시오.

2. WHERE(조건문)절과 ORDER BY(정렬)절

1. WHERE절

1) 리터럴 문자

- 리터럴 문자란? 쿼리구문에 포함된 일반 문자, 숫자, 날짜 값
- 문자나 날짜 리터럴은 작은 따옴표로 묶어서 작성해야함.
- MySQL은 작은따옴표, 큰따옴표 모두 허용하나 다른 DBMS가 작은따옴표만 허용하는 경우가 많으니 작은따옴표로 기억하는 것을 권장함.
- 문자 : MySQL은 대소문자 구분하지 않음. (DBMS에 따라 다름)
- 날짜 : MySQL은 년-월-일 순서로 작성해야함. (DBMS에 따라 다름)

2) WHERE절이 포함된 SQL구문 예제

```
SELECT employee_id, last_name, job_id, department_id
FROM employees
WHERE department_id = 90 ;
```

```
SELECT last_name, job_id, department_id
FROM employees
WHERE last_name = 'Whalen' ;
```

```
SELECT last_name
FROM employees
WHERE hire_date = '1996-02-17' ;
```

```
SELECT last_name, salary
FROM employees
WHERE salary <= 3000 ;
```

[between A and B 비교연산자]

```
SELECT last_name, salary
FROM employees
WHERE salary BETWEEN 2500 AND 3500 ;
```

[in 비교연산자(다중행비교연산자)]

```
SELECT employee_id, last_name, salary, manager_id
FROM employees
WHERE manager_id IN (100, 101, 201) ;
```

MySQL

[like 비교연산자]

```
SELECT last_name
FROM employees
WHERE last_name LIKE '_o%' ;
```

[is null 비교연산자]

```
SELECT last_name, manager_id
FROM employees
WHERE manager_id IS NULL ;
```

[where절에 여러 조건문 작성하기 - AND, OR 논리연산자]

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary >= 10000
AND job_id LIKE '%MAN%' ;
```

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary >= 10000
OR job_id LIKE '%MAN%' ;
```

- 비교연산자에 NOT(논리연산자)이 조합으로 사용된 경우


```
... WHERE job_id NOT IN ('AC_ACCOUNT', 'AD_VP')
... WHERE salary NOT BETWEEN 10000 AND 15000
... WHERE last_name NOT LIKE '%A%'
... WHERE commission_pct IS NOT NULL
```

MySQL

2. ORDER BY절


- ASC: Ascending order, default
- DESC: Descending order

```
SELECT last_name, job_id, department_id, hire_date
FROM employees
ORDER BY hire_date DESC ;
```

1


[order by절에 column alias 사용]

```
SELECT employee_id, last_name, salary*12 annsal
FROM employees
ORDER BY annsal ;
```

2


[order by절에 위치표기법 사용]

```
SELECT last_name, job_id, department_id, hire_date
FROM employees
ORDER BY 3 ;
```

3

[다중 컬럼을 기준으로 정렬하기]

```
SELECT last_name, department_id, salary
FROM employees
ORDER BY department_id, salary DESC ;
```

4

<연습문제>

1. employees 테이블로부터 2000년도에 입사한 모든 사원의 last_name과 hire_date를 출력하시오.
2. employees 테이블로부터 커미션을 받지 않는 모든 사원의 last_name, salary, commission_pct를 출력하되 salary를 기준으로 내림차순 정렬하시오.

3. HR 스키마 - ERD 만들기

1. ERD란?

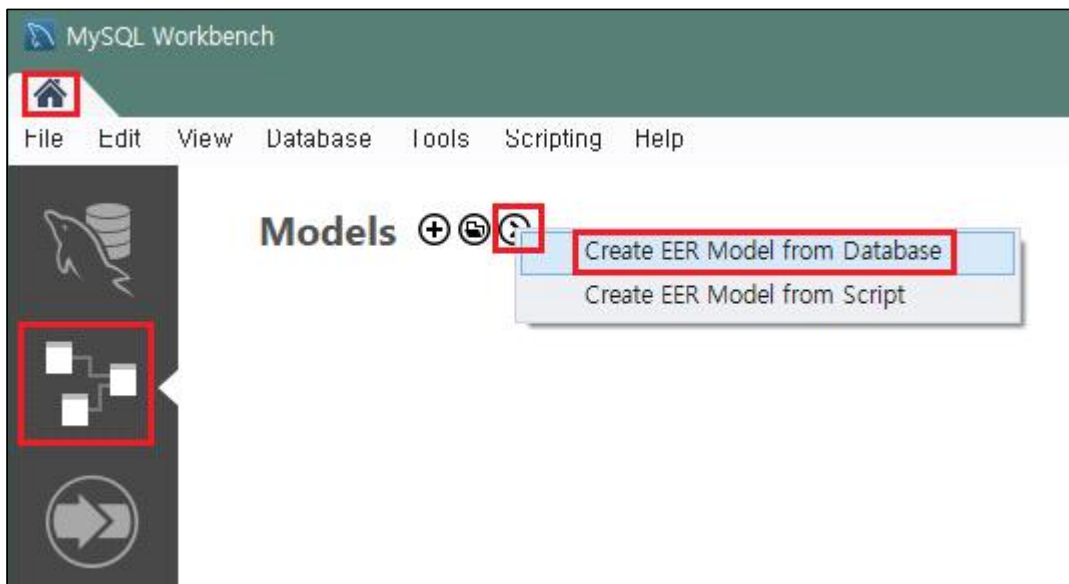
- 개체-관계 다이어그램(Entity-Relationship Diagram)의 약자로 개체(쉽게 테이블을 생각하면 됨)와 개체 사이의 관계를 도표(다이어그램)로 표현하는 방법
- 데이터모델링 단원에서 자세히 소개될 예정임.

2. 리버스 모델링

- 기존의 Database(Schema)를 ERD로 바꾸는 작업
- Database의 전체적인 구조를 파악하기 쉬움.

3. HR 스키마 → ERD

1) MySQL Workbench [Home] - [Models] - [Create EER Model from Database]



MySQL

2) 접속 정보 입력 후 [Next]

The screenshot shows the 'Set Parameters for Connecting to a DBMS' dialog box. On the left is a sidebar with 'Connection Options' selected. The main area has tabs for 'Parameters', 'SSL', and 'Advanced'. Under 'Parameters', the 'Stored Connection' is set to 'Local Instance MySQL', and the 'Connection Method' is 'Standard (TCP/IP)'. The 'Hostname' is 'localhost' and the 'Port' is '3306'. The 'Username' is 'root'. The 'Password' field has 'Store in Vault ...' and 'Clear' buttons. At the bottom are 'Back', 'Next', and 'Cancel' buttons.

Reverse Engineer Database

Connection Options

- Connect to DBMS
- Select Schemas
- Retrieve Objects
- Select Objects
- Reverse Engineer
- Results

Set Parameters for Connecting to a DBMS

Stored Connection: Local Instance MySQL Select from saved connection settings

Connection Method: Standard (TCP/IP) Method to use to connect to the RDBMS

Parameters SSL Advanced

Hostname: localhost Port: 3306 Name or IP address of the server host - and TCP/IP port.

Username: root Name of the user to connect with.

Password: Store in Vault ... Clear The user's password. Will be requested later if it's not set.

Back Next Cancel

3) DBMS 접속이 완료되면 [Next]

The screenshot shows the 'Connect to DBMS and Fetch Information' dialog box. The sidebar has 'Connect to DBMS' selected. The main area shows a list of tasks to be executed: 'Connect to DBMS', 'Retrieve Schema List from Database', and 'Check Common Server Configuration Issues'. Below the list, it says 'Execution Completed Successfully' and 'Fetch finished.' At the bottom are 'Show Logs', 'Back', 'Next', and 'Cancel' buttons.

Reverse Engineer Database

Connect to DBMS

The following tasks will now be executed. Please monitor the execution. Press Show Logs to see the execution logs.

- ☒ Connect to DBMS
- ☒ Retrieve Schema List from Database
- ☒ Check Common Server Configuration Issues

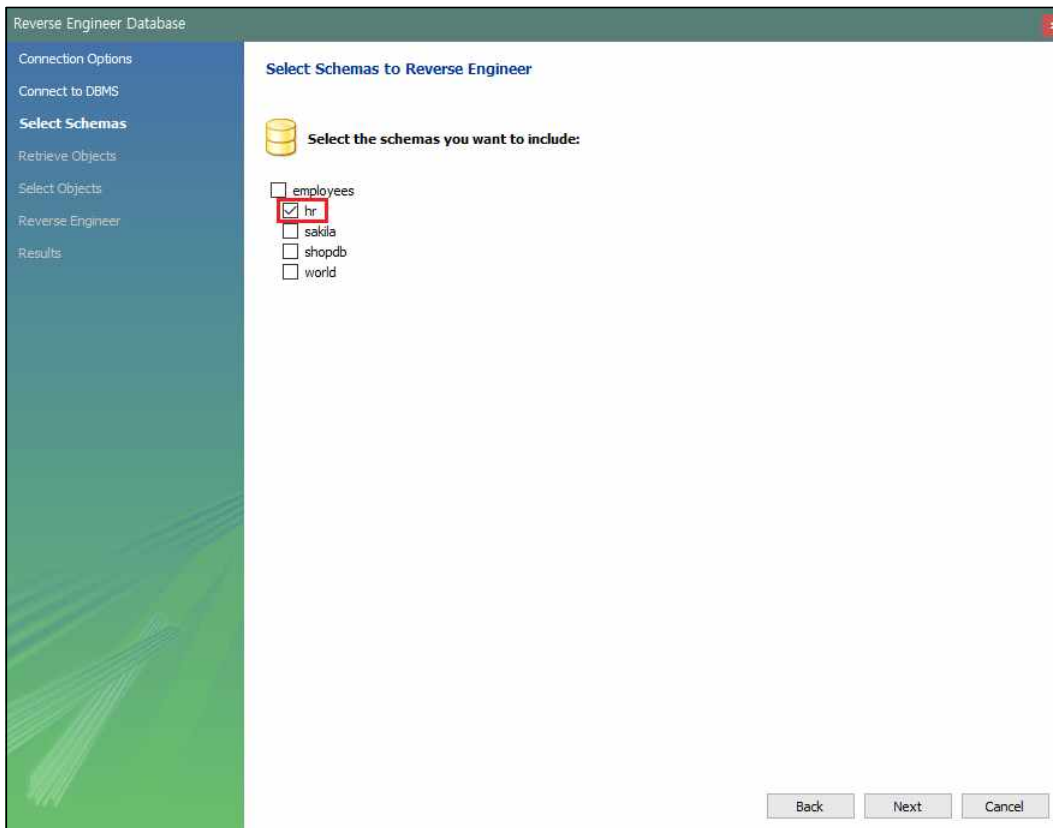
Execution Completed Successfully

Fetch finished.

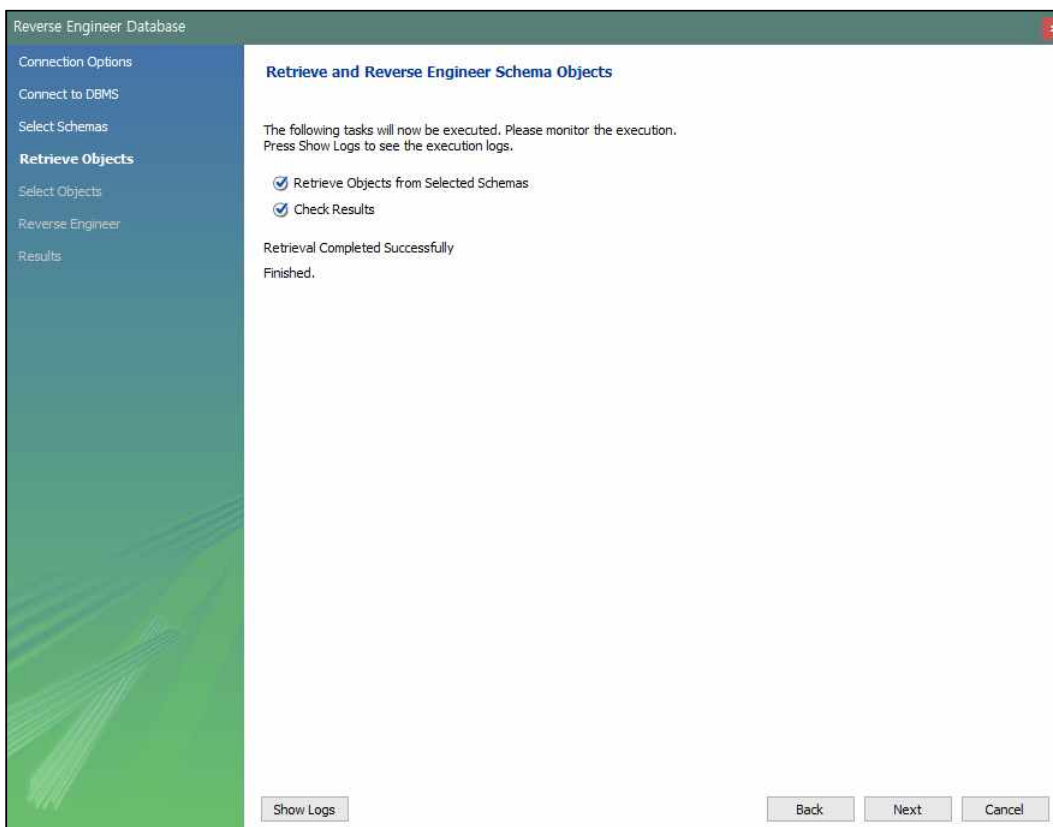
Show Logs Back Next Cancel

MySQL

4) ERD를 생성할 기존 Database(Schema) 선택 후 [Next]

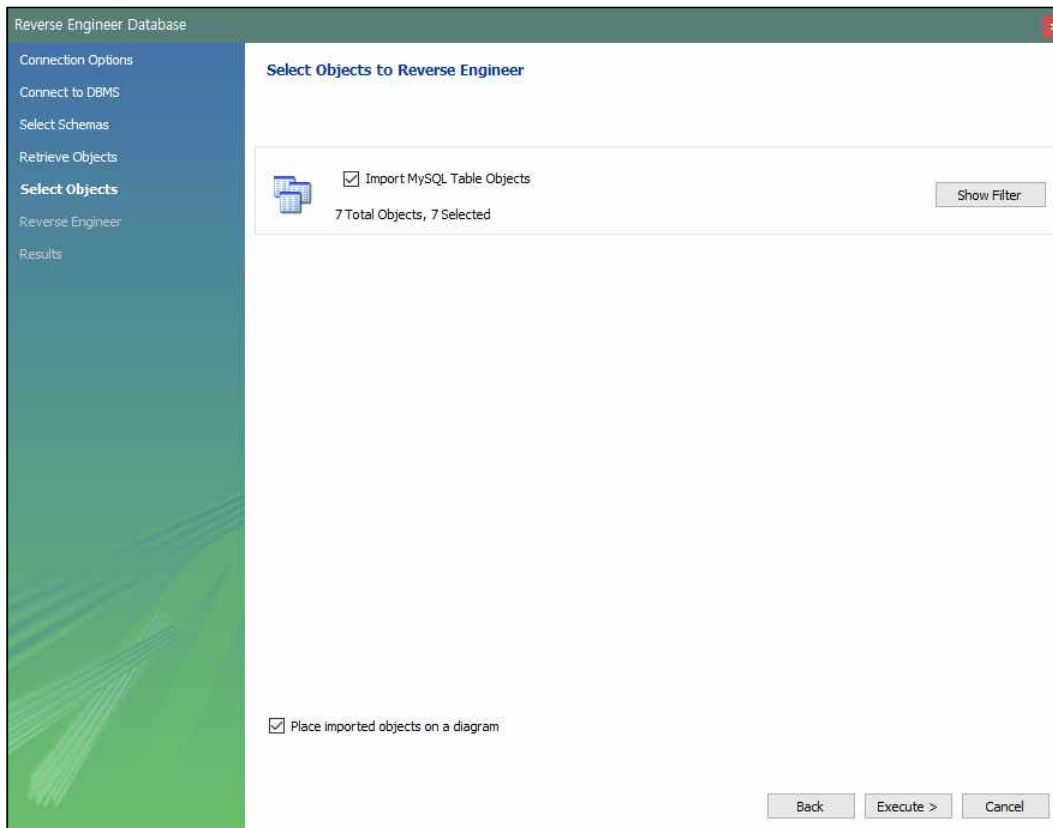


5) 선택한 스키마 확인 작업이 완료되면 [Next]



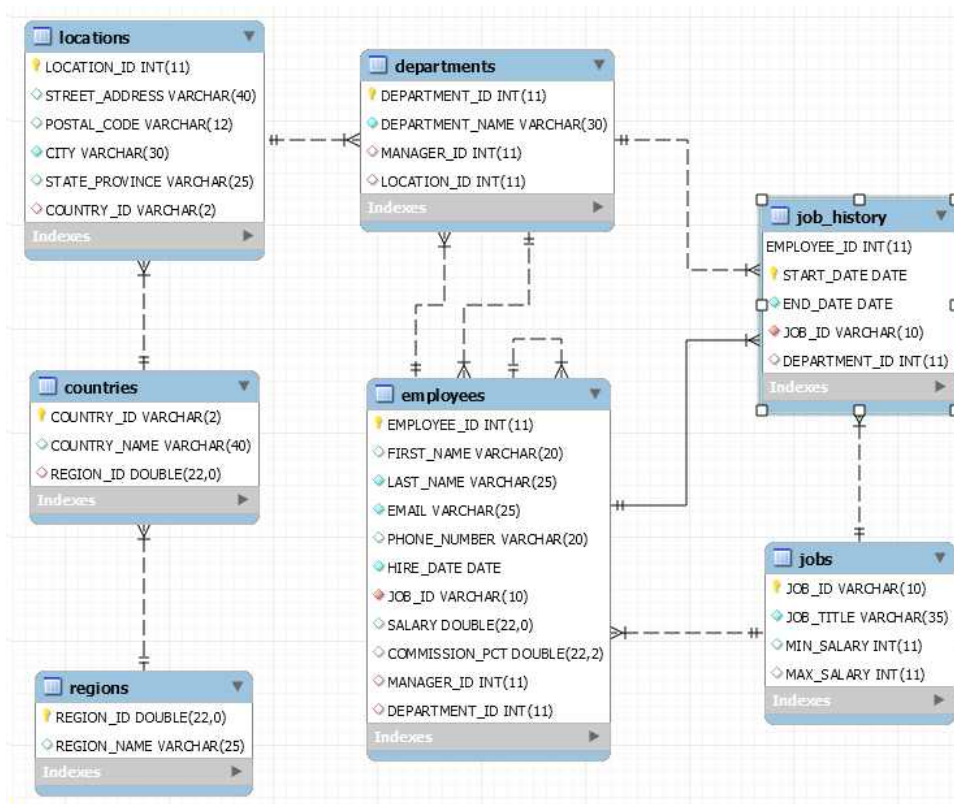
MySQL

6) 리버스 엔지니어링(역모델링) 작업을 할 테이블 확인 후 [Execute]



7) 리버스 엔지니어링 작업이 완료되면 [Next] 후 [Finish]

8) ERD 확인하기



4. JOIN

1. ON절을 사용한 JOIN

```
SELECT e.employee_id, e.last_name, e.department_id,  
       d.department_id, d.location_id  
FROM   employees e JOIN departments d  
ON     (e.department_id = d.department_id);
```

```
SELECT e.employee_id, e.last_name, e.department_id,  
       d.department_id, d.location_id  
FROM   employees e JOIN departments d  
ON     (e.department_id = d.department_id)  
WHERE  e.manager_id = 149 ;
```

2. ON절을 사용한 Self-Join

```
SELECT worker.last_name emp, manager.last_name mgr  
FROM   employees worker JOIN employees manager  
ON     (worker.manager_id = manager.employee_id);
```

<연습문제>

1. employees 테이블과 departments 테이블을 조인하여 모든 사원의 정보와 함께 부서 정보를 함께 출력하시오.

출력형식 :

employee_id	last_name	salary	department_id	department_name
-------------	-----------	--------	---------------	-----------------

2. employees 테이블로부터 모든 사원의 last_name, employee_id, 매니저 이름, manager_id를 함께 출력하시오.

출력형식 :

Employee	Emp#	Manager	Mgr#
(last_name)	(employee_id)	(매니저이름)	(manager_id)

5-1. 단일 행 함수 - 문자함수

1. 문자함수

ASCII(아스키코드)	문자의 아스키 코드값을 반환해 주는 함수
CHAR(숫자)	숫자의 아스키 코드값을 반환해 주는 함수
예제	<code>select ascii('A'), char(65);</code>

※ MySQL에서는 단순한 함수 결과나 계산 결과를 출력하는 경우 즉, 접근해야하는 테이블이 없는 출력 구문인 경우 select절만 작성할 수 있음.

※ 출력 결과에 BLOB로 표시되는 경우(Workbench의 버그) 오른쪽 마우스 클릭 후 [Open Value in Viewer] 선택 후 [Text] 탭에 들어가면 정상적인 출력 결과를 확인할 수 있음.

LENGTH(문자열)	문자열의 byte 수를 반환해 주는 함수
BIT_LENGTH(문자열)	할당된 bit 크기 또는 문자 크기를 반환해 주는 함수
CHAR_LENGTH(문자열)	문자의 개수를 반환해 주는 함수
예제	<code>select length('abc'), bit_length('abc'), char_length('abc'); select length('가나다'), bit_length('가나다'), char_length('가나다');</code>

CONCAT(문자열1, 문자열2, ...)	문자열을 연결해 주는 함수
CONCAT_WS(구분자, 문자열1, 문자열2, ...)	구분자와 함께 문자열을 연결해 주는 함수
예제	<code>select employee_id, concat(first_name, last_name) as "이름" from employees; select employee_id, concat(first_name, ' ', last_name) as "이름" from employees; select concat_ws('/', '2025', '01', '01'); select concat_ws('--', last_name, job_id, salary) from employees;</code>

INSTR(기준 문자열, 부분 문자열)	기준 문자열에서 부분 문자열의 시작 위치값을 반환해 주는 함수
예제	<code>select instr('하나둘셋', '둘'); select last_name, instr(last_name, 'a') from employees;</code>

MySQL

UPPER(문자열)	문자열을 대문자로 변환해 주는 함수
LOWER(문자열)	문자열을 소문자로 변환해 주는 함수
예제	<pre>select lower('abcdEFGH'), upper('abcdEFGH');</pre> <pre>select employee_id, upper(last_name) as "L-name", lower(job_id) as "Job", phone_number, lower(email) as "E-mail" from employees;</pre> <pre>select concat('The job id for ' , upper(last_name) , ' is ' , lower(job_id)) from employees;</pre>

LEFT(문자열, 길이)	왼쪽에서 문자열의 길이만큼 반환해 주는 함수
RIGHT(문자열, 길이)	오른쪽에서 문자열의 길이만큼 반환해 주는 함수
예제	<pre>select left('abcdefghi', 3), right('abcdefghi', 3);</pre> <pre>select last_name, left(last_name, 2), right(last_name, 2) from employees;</pre>

LPAD(문자열, 길이, 채울 문자)	문자열을 길이만큼 늘린 후 빈곳을 왼쪽부터 채울 문자로 채워주는 함수, 오른쪽 정렬 함수
RPAD(문자열, 길이, 채울 문자)	문자열을 길이만큼 늘린 후 빈곳을 오른쪽부터 채울 문자로 채워주는 함수, 왼쪽 정렬 함수
예제	<pre>select lpad('가나다', 5, '#'), rpad('가나다', 5, '#');</pre> <pre>select lpad(last_name, 20, '_') as "L-name", rpad(first_name, 20, '_') as "F-name" from employees;</pre>

LTRIM(문자열)	문자열의 왼쪽 공백을 제거해 주는 함수
RTRIM(문자열)	문자열의 오른쪽 공백을 제거해 주는 함수
예제	<pre>select ltrim(' SQL 문법 '), rtrim(' SQL 문법 ');</pre>

※ 중간 공백은 제거되지 않음.

MySQL

TRIM(문자열)	문자열의 앞/뒤 공백을 제거해 주는 함수
TRIM(방향 자를문자열 FROM 문자열)	방향 : leading(앞), trailing(뒤), both(양쪽) 문자열로부터 해당 방향의 자를 문자열을 제거해 주는 함수
예제	select trim(' SQL 문법 '), trim(both ' ' from ' __SQL_문법__');

REPLACE(문자열, 기존문자열, 바꿀문자열)	문자열에 기존 문자열을 바꿀 문자열로 교체해 주는 함수
예제	select employee_id, phone_number, replace(phone_number, ',', '-') as "전화번호" from employees;

SPACE(길이)	길이만큼의 공백을 반환해 주는 함수
예제	select concat('MySQL', space(10), 'DBMS');

SUBSTR(문자열, 시작위치, 길이)	문자열의 일부분을 반환해 주는 함수
예제	select substr('대한민국만세', 3, 2); select last_name, substr(last_name, 2, 3), substring(last_name, 2, 3) from employees;

MySQL

<연습문제>

1. employees 테이블로부터 직원들의 last_name과 last_name의 길이를 출력하되 last_name이 'J', 'A', 'M'으로 시작되는 직원만 출력하시오. 또한 last_name을 기준으로 오름차순 정렬해서 출력하시오.

	Name	Length
1	Abel	4
2	Matos	5
3	Mourgos	7

2. employees 테이블로부터 직원들의 last_name과 salary를 출력하되 특히 급여는 15자리로 표시하고 왼쪽부터 \$ 기호가 채워지도록 지정하시오.

	LAST_NAME	SALARY
1	Whalen	\$\$\$\$\$\$\$\$\$4400
2	Hartstein	\$\$\$\$\$\$\$\$\$13000
3	Fay	\$\$\$\$\$\$\$\$\$6000
4	Higgins	\$\$\$\$\$\$\$\$\$12000
5	Gietz	\$\$\$\$\$\$\$\$\$8300

3. employees 테이블로부터 last_name과 급여 액수를 별표(*)로 나타내는 query를 작성하시오.

각 별표는 \$1,000를 의미하며 백단위 이하는 표시하지 않습니다. 또한 급여의 내림차순으로 데이터를 정렬하여 출력하고 컬럼 제목(column alias)을 EMPLOYEES_AND_THEIR_SALARIES로 지정하시오.

	EMPLOYEES_AND_THEIR_SALARIES
1	King *****
2	Kochhar *****
3	De Haan *****
4	Hartstei *****
5	Higgins *****

5-2. 단일 행 함수 - 숫자함수, 날짜함수

1. 숫자함수

ROUND(숫자, 반올림할 자리)	숫자를 반올림할 자리까지 반올림을 해주는 함수 반올림할 자리 생략 시 <u>일의 자리</u> 로 반올림함
TRUNCATE(숫자, 버림할 자리)	숫자를 버림할 자리까지 남기고 버림을 해주는 함수
예제	select round(45.923, 2), round(45.923, 0), round(45.923, -1); select truncate(45.923, 2), truncate(45.923, 0), truncate(45.923, -1);

CEIL(숫자)	숫자보다 크거나 같은 최소의 정수를 반환해 주는 함수 <u>일의 자리</u> 로 올림을 해주는 함수
FLOOR(숫자)	숫자보다 작거나 같은 최소의 정수를 반환해 주는 함수 <u>일의 자리</u> 까지 버림을 해주는 함수
예제	select ceil(45.923), ceil(52.1); select floor(45.923), floor(52.1);

MOD(숫자1, 숫자2)	숫자1을 숫자2로 나눈 나머지를 반환해 주는 함수
예제	select mod(157, 10), 157 mod 10, 157 % 10, 157 / 10; select last_name, salary, mod(salary, 5000) from employees where job_id = 'SA_REP'; // 값이 짝수인지 홀수인지를 확인하는 용도로도 활용됨. select employee_id, mod(employee_id, 2) from employees;

ABS(숫자)	숫자의 절대 값을 반환해 주는 함수
예제	select abs(-5), abs(5), abs(-4.5);

POWER(숫자, 제곱값) POW(숫자, 제곱값)	숫자의 제곱 값을 계산하여 반환해 주는 함수
예제	select power(2, 3), power(8, 3); select power(4, 1), power(4, 2), power(4, 3), power(4, 4), power(4, 5);

MySQL

SIGN(숫자)	숫자가 양수이면 1, 음수이면 -1, 0이면 0을 반환해 주는 함수
예제	select sign(3), sign(-3), sign(4.26), sign(-4.26), sign(0);

2. 날짜함수

NOW()	현재 날짜와 시간을 반환해 주는 함수 (년/월/일/시/분/초)
SYSDATE()	
CURRENT_TIMESTAMP()	
예제	select now(); select sysdate(); select current_timestamp();

CURRENT_DATE() CURDATE()	현재 날짜를 반환해 주는 함수(년/월/일)
CURRENT_TIME() CURTIME()	현재 시간을 반환해 주는 함수(시/분/초)
예제	select current_date(); select current_time();

YEAR(날짜)	날짜/시간에서 년도를 반환해 주는 함수
MONTH(날짜)	날짜/시간에서 월을 반환해 주는 함수
DAY(날짜) DAYOFMONTH(날짜)	날짜/시간에서 일을 반환해 주는 함수
HOUR(시간)	날짜/시간에서 시간을 반환해 주는 함수
MINUTE(시간)	날짜/시간에서 분을 반환해 주는 함수
SECOND(시간)	날짜/시간에서 초를 반환해 주는 함수
예제	select year(now()), month(now()), day(now()), hour(now()), minute(now()), second(now()); select last_name, hire_date, year(hire_date), month(hire_date), day(hire_date) from employees where department_id = 90;

MySQL

DATE(날짜와 시간)	날짜/시간에서 날짜를 반환해 주는 함수(년/월/일)
TIME(날짜와 시간)	날짜/시간에서 시간을 반환해 주는 함수(시/분/초)
예제	<code>select date(now()), time(now());</code>

ADDDATE(날짜, 차이) DATE_ADD(날짜, 차이)	날짜에서 차이를 더한 날짜를 반환해 주는 함수
SUBDATE(날짜, 차이) DATE_SUB(날짜, 차이)	날짜에서 차이를 뺀 날짜를 반환해 주는 함수
예제	<pre>select adddate('2022-01-01', interval 35 day), adddate('2022-01-01', interval 2 month), date_add('2022-01-01', interval 1 year); select subdate('2022-01-01', interval 35 day), subdate('2022-01-01', interval 2 month), date_sub('2022-01-01', interval 1 year); select last_name, hire_date, adddate(hire_date, interval 6 month) as "입사 6개월 후", subdate(hire_date, interval 7 day) as "입사 7일전" from employees where department_id = 60;</pre>

ADDTIME(날짜와 시간, 시간)	날짜/시간에서 시간을 더한 결과를 반환하는 함수
SUBTIME(날짜와 시간, 시간)	날짜/시간에서 시간을 뺀 결과를 반환하는 함수
예제	<pre>select addtime('2022-01-01 23:59:59', '1:1:1'), addtime('15:00:00', '2:10:10'); select subtime('2022-01-01 23:59:59', '1:1:1'), subtime('15:00:00', '2:10:10');</pre>

MySQL

DATEDIFF(날짜1, 날짜2)	날짜1 - 날짜2를 반환하는 함수
TIMEDIFF(시간1, 시간2)	시간1 - 시간2를 반환하는 함수
예제	<pre>select datediff('2022-12-31', now()), datediff(now(), '2022-12-31'); select timediff('23:23:59', '12:11:10'), timediff('12:11:10', '23:23:59'); select last_name, hire_date, datediff(now(), hire_date) as "근무한 일수", from employees;</pre>

DAYOFWEEK(날짜)	날짜의 요일을 반환해 주는 함수 (1-일, 2-월, 3-화, 4-수, 5-목, 6-금, 7-토)
MONTHNAME(날짜)	날짜의 월의 영문 이름을 반환해 주는 함수
DAYOFYEAR(날짜)	날짜가 1년 중 몇 번째 날짜인지를 반환해 주는 함수
예제	<pre>select dayofweek(now()), monthname(now()), dayofyear(now()); select employee_id, last_name, hire_date, monthname(hire_date) from employees;</pre>

LAST_DAY(날짜)	날짜가 속한 월의 마지막 날짜를 반환하는 함수 주로 그 달이 몇일까지 있는지 확인할 때 사용함
예제	<pre>select last_day('2022-04-03'); select employee_id, last_name, hire_date, last_day(hire_date) from employees;</pre>

QUARTER(날짜)	날짜가 4분기 중에서 몇 분기인지를 반환하는 함수
예제	<pre>select quarter('2022-01-31'), quarter('2022-02-05'), quarter('2022-03-15'), quarter('2022-04-02'), quarter('2022-05-10'), quarter('2022-06-25'), quarter('2022-07-10'), quarter('2022-08-09'), quarter('2022-09-16'), quarter('2022-10-14'), quarter('2022-11-11'), quarter('2022-12-25');</pre>

<연습문제>

1. employees 테이블로부터 전 사원의 employee_id, last_name, salary를 출력하고 마지막 컬럼에는 15.5% 인상된 급여(일의 자리에서 반올림)를 New salary라는 제목으로 출력하는 구문을 작성하시오.

	EMPLOYEE_ID	LAST_NAME	SALARY	NewSalary
1	200	Whalen	4400	5082
2	201	Hartstein	13000	15015
3	202	Fay	6000	6930
4	205	Higgins	12000	13860
5	206	Gietz	8300	9587

...

2. employees 테이블로부터 사원들의 employee_id, last_name, salary, 15.5% 인상된 급여(New Salary), 새 급여에서 이전 급여를 뺀 값(Increase)을 출력하는 구문을 작성하시오.
(단, 4번째, 5번째 컬럼은 일의 자리에서 반올림하여 정수로 표현하시오.)

	EMPLOYEE_ID	LAST_NAME	SALARY	NewSalary	Increase
1	200	Whalen	4400	5082	682
2	201	Hartstein	13000	15015	2015
3	202	Fay	6000	6930	930
4	205	Higgins	12000	13860	1860
5	206	Gietz	8300	9587	1287

3. employees 테이블로부터 2월에 입사한 사원들의 employee_id, last_name, job_id, hire_date, department_id를 출력하시오.

4. employees 테이블로부터 1990년부터 1995년에 입사한 사원들의 employee_id, last_name, hire_date, salary, department_id를 출력하시오.

5. employess 테이블로부터 오늘 날짜를 기준으로 근무한 주수가 1200주 미만인 사원들의 last_name, hire_date, 근무한 일수, 근무한 주수를 출력하는 구문을 작성하시오.

6. employees 테이블로부터 전 사원들의 employee_id, last_name, hire_date, 입사한 날짜에 해당되는 분기를 출력하는 구문을 작성하시오.

특히 입사한 날짜에 해당되는 분기를 출력해야하는 4번째 컬럼은 예를 들어 입사일이 2000-05-05인 경우 2분기라고 출력될 수 있도록 작성하시오.

<출력형식 예제>

employee_id	last_name	hire_date	입사한 분기
100	King	2000-05-05	2분기
101	DeHaan	1998-12-25	4분기

5-3. 단일 행 함수 - 변환함수, 제어 흐름 함수, 시스템 정보 함수

1. 변환함수

DATE_FORMAT(날짜, 형식)	날짜를 형식에 맞게 출력하는 함수
예제	<pre>select date_format(now(), '%Y-%M-%d') as "Now";</pre> <pre>select date_format(now(), '%Y/%M/%d %H:%i:%s') as "Now";</pre> <pre>select employee_id, date_format(hire_date, '%Y-%M-%d %W') as "입사일" from employees;</pre>

Format	설명
%Y	년도를 4자리 숫자로 표현
%y	년도를 2자리 숫자로 표현
%M	월을 영문 풀네임으로 표현
%b	월을 영문 약자로 표현
%m	월을 2자리 숫자로 표현
%c	월을 2자리 숫자로 표현하되 10보다 작을 경우 한자리로 표현
%d	일을 2자리 숫자로 표현
%e	일을 2자리 숫자로 표현하되 10보다 작을 경우 한자리로 표현
%W	요일을 영문 풀네임으로 표현
%a	요일을 영문 약자로 표현
%H	시간을 24시간으로 표현
%h	시간을 12시간으로 표현
%k	시간을 24시간으로 표현하되 10보다 작을 경우 한자리로 표현
%l (소문자 엘)	시간을 12시간으로 표현하되 10보다 작을 경우 한자리로 표현
%i	분을 2자리 숫자로 표현
%s	초를 2자리 숫자로 표현

CAST(값 AS 데이터타입)		값을 지정된 데이터타입으로 변환하는 함수
예제	<pre> select cast("123" as signed), cast("-123.45" as signed); select cast("123" as unsigned), cast("123.45" as unsigned); select cast('2022/03/18' as date) as '날짜'; select cast('2022@03@18' as date) as '날짜'; select cast('2022-01-02 21:24:33.123' as date) as "DATE", cast('2022-01-02 21:24:33.123' as time) as "TIME", cast('2022-01-02 21:24:33.123' as datetime) as "DATETIME"; </pre>	

데이터타입	데이터타입
BINARY	CHAR (문자형)
SIGNED (부호 있는 정수형)	UNSIGNED (부호 없는 정수형)
DECIMAL (숫자형)	DOUBLE (숫자형)
FLOAT (숫자형)	DATETIME (날짜형)
DATE (날짜형)	TIME (시간)

2. 제어 흐름 함수

IF(논리식, 참일 때 값, 거짓일 때 값)		논리식이 참이면 참일 때 값을 출력하고 거짓이면 거짓일 때 값을 출력하는 함수
예제	<pre> select if(100>200, '참이다', '거짓이다') as "결과"; select employee_id, salary, if(salary>10000, '1등급', '2등급') as "급여 등급" from employees; </pre>	

IFNULL(수식1, 수식2)		수식1이 NULL이 아니면 수식1이 반환되고, 수식1이 NULL이면 수식2가 반환되는 함수
예제	<pre> select ifnull(null, '널이군요') as "결과1", ifnull(100, '널이군요') as "결과2"; select employee_id, last_name, salary, commission_pct, ifnull(commission_pct, 0) from employees; </pre>	

MySQL

NULLIF(수식1, 수식2)	수식1과 수식2가 같으면 NULL을 반환하고, 다르면 수식1을 반환하는 함수
예제	<pre>select nullif(100, 100) as "결과1", nullif(100, 200) as "결과2"; select employee_id, first_name, last_name, nullif(length(first_name), length(last_name)) as "결과" from employees;</pre>

CASE 비교값 WHEN 값1 THEN 결과1 WHEN 값2 THEN 결과2 ... ELSE 기본값 END	SQL구문에서 if-then-else의 논리를 적용할 수 있는 연산자 함수는 아니나 제어 흐름 함수와 함께 정리해 두기
예제	<pre>select case 10 when 1 then '일' when 5 then '오' when 10 then '십' else '모름' end as "case예제"; select employee_id, last_name, department_id, case department_id when 10 then '부서 10' when 50 then '부서 50' when 100 then '부서 100' when 150 then '부서 150' when 200 then '부서 200' else '기타 부서' end as "부서 정보" from employees;</pre>

MySQL

3. 시스템 정보 함수

USER() CURRENT_USER() SESSION_USER()	현재 사용자 정보를 반환하는 함수
DATABASE() SCHEMA()	현재 데이터베이스 또는 스키마 정보를 반환하는 함수
VERSION()	현재 MySQL 버전을 반환하는 함수
예제	<pre>select user(), current_user(), session_user(); select database(), schema(); select version();</pre>

<연습문제>

1. employees 테이블로부터 직원들의 last_name과 commission_pct를 출력하되 커미션을 받는 직원은 자신의 커미션 비율을 출력하고, 커미션을 받지 않는 직원은 "No Commission"을 출력하는 구문을 작성하시오.

R	LAST_NAME	R	COMM
1	Whalen		No Commission
2	Hartstein		No Commission
3	Fay		No Commission
4	Higgins		No Commission
5	Gietz		No Commission

...

16	Vargas		No Commission
17	Zlotkey		.2
18	Abel		.3
19	Taylor		.2
20	Grant		.15

2. employees 테이블로부터 JOB_ID 값을 기반으로 모든 직원의 등급을 표시하는 query를 작성하시오.

[조건표]

job_id	등급
AD_PRES	A
ST_MAN	B
IT_PROG	C
SA_REP	D
ST_CLERK	E
나머지	0

[결과]

R	JOB_ID	R	GRADE
1	AC_ACCOUNT		0
2	AC_MGR		0
3	AD_ASST		0
4	AD_PRES		A
5	AD_VP		0
6	AD_VP		0
7	IT_PROG		C

6. 그룹 함수와 그룹화

1. 그룹함수

MIN(행그룹)	행그룹에서 최소값을 구해주는 함수 모든 데이터타입에 사용 가능함.
MAX(행그룹)	행그룹에서 최대값을 구해주는 함수 모든 데이터타입에 사용 가능함.
예제	<pre>select min(salary) as "최소 급여", max(salary) as "최대 급여" from employees;</pre> <pre>select min(hire_date) as "가장 오래된 입사일", max(hire_date) as "가장 최근 입사일" from employees;</pre> <pre>select min(last_name) as "name1", max(last_name) as "name2" from employees;</pre>

SUM(행그룹)	행그룹의 합계를 구해주는 함수
AVG(행그룹)	행그룹의 평균을 구해주는 함수
예제	<pre>select sum(salary) as "급여 합계", avg(salary) as "평균 급여" from employees;</pre> <pre>select sum(salary) as "급여 합계", avg(salary) as "평균 급여" from employees where job_id like '%REP%';</pre>

COUNT(*)	행그룹에서 행의 개수를 반환해주는 함수 (null값 포함, 중복값 포함)
COUNT(행그룹)	행그룹에서 행의 개수를 반환해주는 함수 (null값 제외, 중복값 포함)
COUNT(DISTINCT 행그룹)	행그룹에서 행의 개수를 반환해주는 함수 (null값 제외, 중복값 제외)
예제	<pre> select count(*) from employees; //전 직원 수 select count(commission_pct) from employees; //커미션을 받는 직원 수 ----- select count(*) from employees where department_id <= 80; //80번 이하 부서에 소속된 직원 수 select count(commission_pct) from employees //80번 이하 부서에 소속된 직원 중 커미션을 where department_id <= 80; 받는 직원 수 ----- select count(department_id) from employees; select count(distinct department_id) from employees; </pre>

2. GROUP BY절

※ 그룹함수와 group by절 사용 시 규칙(문법)

=> select절의 컬럼리스트들 중 그룹함수에 포함된 컬럼과 그룹함수에 포함되지 않은 컬럼이 같이 출력되려면 그룹함수에 포함되지 않은 컬럼은 반드시 group by절에 포함되어 있어야 함!

```

SELECT    department_id, AVG(salary)
FROM      employees
GROUP BY department_id ;

```

MySQL

```
SELECT    department_id, job_id, SUM(salary)
FROM      employees
WHERE     department_id > 40
GROUP BY  department_id, job_id
ORDER BY  department_id;
```

[오류의 원인 파악 및 수정하기]

```
SELECT department_id, COUNT(last_name)
FROM   employees;
```

ORA-00937: not a single-group group function
00937. 00000 - "not a single-group group function"

A GROUP BY clause must be added to count the last names for each department_id.

```
SELECT department_id, job_id, COUNT(last_name)
FROM   employees
GROUP BY department_id;
```

ORA-00979: not a GROUP BY expression
00979. 00000 - "not a GROUP BY expression"

Either add job_id in the GROUP BY or remove the job_id column from the SELECT list.

3. HAVING절

- WHERE절 : 행 제한 조건절
- HAVING절 : 행그룹 제한 조건절

```
SELECT    job_id, SUM(salary) PAYROLL
FROM      employees
WHERE     job_id NOT LIKE '%REP%'
GROUP BY  job_id
HAVING    SUM(salary) > 13000
ORDER BY  SUM(salary);
```

<연습문제>

- employees 테이블로부터 전체 직원들의 커미션 평균을 출력하는 구문을 작성하시오.
단, 소수점 둘째자리까지 반올림해서 출력하시오.

avg_comm
0.03

- employees 테이블로부터 업무(job_id)별 최대 급여(Maximum), 최소 급여(Minimum), 급여의 합계(Sum), 평균 급여(Average)를 출력하시오.

R	JOB_ID	Maximum	Minimum	Sum	Average
1	AC_MGR	12000	12000	12000	12000
2	AC_ACCOUNT	8300	8300	8300	8300
3	IT_PROG	9000	4200	19200	6400
4	ST_MAN	5800	5800	5800	5800
5	AD_ASST	4400	4400	4400	4400
6	AD_VP	17000	17000	34000	17000
7	MK_MAN	13000	13000	13000	13000
8	SA_MAN	10500	10500	10500	10500
9	MK_REP	6000	6000	6000	6000
10	AD_PRES	24000	24000	24000	24000
11	SA_REP	11000	7000	26600	8867
12	ST_CLERK	3500	2500	11700	2925

- employees 테이블로부터 동일 업무(job_id)를 수행하는 직원 수를 출력하는 구문을 작성하시오.

R	JOB_ID	COUNT(*)
1	AC_ACCOUNT	1
2	AC_MGR	1
3	AD_ASST	1
4	AD_PRES	1
5	AD_VP	2
6	IT_PROG	3
7	MK_MAN	1
8	MK_REP	1
9	SA_MAN	1
10	SA_REP	3
11	ST_CLERK	4
12	ST_MAN	1

MySQL

4. employees 테이블로부터 매니저를 알 수 없는 사원은 제외하고 매니저별로 그룹화하여 매니저별 최소 급여를 출력하되 최소 급여가 \$6000 이상인 그룹만 출력하시오. 또한 최소 급여를 기준으로 내림차순으로 정렬하여 출력하시오.

	MANAGER_ID	MIN(SALARY)
1	102	9000
2	205	8300
3	149	7000

5. employees 테이블에서 최고 급여와 최저 급여의 차이를 출력하는 구문을 작성하시오.

	DIFFERENCE
1	21500

6. employees 테이블로부터 사원의 총 수와 1995년, 1996년, 1997년, 1998년에 채용된 사원의 수를 표시하는 구문을 작성하시오.

	TOTAL	1995	1996	1997	1998
1	20	1	2	2	3

7. Subquery

1. 서브쿼리(Subquery)란?

- 쿼리 구문 안에 또 다시 쿼리 구문이 포함되어 있는 형태
- Group by 절을 제외한 쿼리구문에 사용 가능
- 서브쿼리 유형 : 단일행 서브쿼리, 다중행 서브쿼리

2. 서브쿼리를 사용하는 이유

- Davies란 직원보다 나중에 입사한 사원을 출력하는 구문을 작성하시오.

Main query:



Davies 이후로 채용된 모든 사원의 이름을 식별하십시오.

Subquery:



Davies가 채용된 시기는 언제입니까?

```
SELECT last_name, hire_date
FROM employees
WHERE hire_date > (SELECT hire_date
FROM employees
WHERE last_name = 'Davies');
```

29-JAN-05 ←

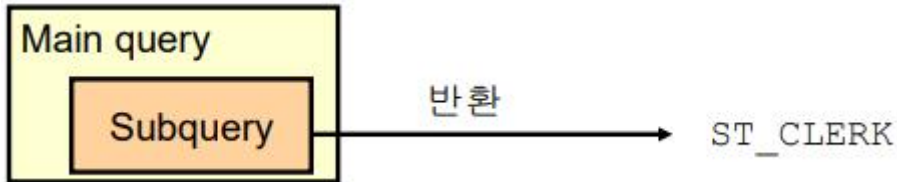
3. 서브쿼리 문법

```
SELECT select_list
FROM table
WHERE expr operator
      (SELECT select_list
FROM table);
```

- Subquery(inner query)를 먼저 실행하여 그 결과를 main query(outer query)에 사용함.
- 서브쿼리는 괄호로 묶어서 작성한다.
- where절 또는 having절에 사용된 경우 가독성을 위해 연산자의 오른쪽에 배치한다.

MySQL

4. 단일행 서브쿼리(Single-row subquery)



- 서브쿼리로부터 메인쿼리로 한 행만 반환되는 유형
- 단일행 서브쿼리인 경우 메인쿼리에 단일행 비교연산자를 사용해야함.

연산자	의미
=	같음
>	보다 큼
>=	보다 크거나 같음
<	보다 작음
<=	보다 작거나 같음
<>	같지 않음

- 단일행 서브쿼리 예제

```
SELECT last_name, salary
FROM employees
WHERE salary >
      (SELECT salary
       FROM employees
       WHERE last_name = 'Abel');
```

```
SELECT last_name, job_id
FROM employees
WHERE job_id =
      (SELECT job_id
       FROM employees
       WHERE employee_id = 141);
```

```
SELECT last_name, job_id, salary
FROM employees
WHERE salary =
      (SELECT MIN(salary)
       FROM employees);
```

```
SELECT last_name, job_id, salary
FROM employees
WHERE job_id = (SELECT job_id
                FROM employees
                WHERE last_name = 'Lee')
AND salary > (SELECT salary
              FROM employees
              WHERE last_name = 'Lee');
```

```
SELECT department_id, MIN(salary)
FROM employees
WHERE department_id is not null
GROUP BY department_id
HAVING MIN(salary) >
        (SELECT MIN(salary)
         FROM employees
         WHERE department_id = 30);
```

- 오류의 원인은?

```
SELECT employee_id, last_name
FROM employees
WHERE salary = (SELECT MIN(salary)
                FROM employees
                GROUP BY department_id);
```

- 결과가 나오지 않는 원인은?

```
SELECT last_name, job_id
FROM employees
WHERE job_id = (SELECT job_id
                FROM employees
                WHERE last_name = 'Haas');
```

MySQL

5. 다중행 서브쿼리(Multiple-row subquery)



- 서브쿼리로부터 메인쿼리로 두 개 이상의 행을 반환되는 유형
- 다중행 서브쿼리인 경우 메인쿼리에 다중행 비교연산자를 사용해야함.

연산자	의미
IN	리스트의 임의 멤버와 같음
ANY	=, !=, >, <, <=, >= 연산자가 앞에 있어야 합니다. 관계가 TRUE인 subquery 의 결과 집합에 요소가 1개 이상 있는 경우 TRUE를 반환합니다.
ALL	=, !=, >, <, <=, >= 연산자가 앞에 있어야 합니다. Subquery 결과 집합의 모든 요소에 대한 관계가 TRUE인 경우 TRUE를 반환합니다.

- 다중행 서브쿼리 예제

```
SELECT employee_id, last_name, manager_id, department_id
FROM employees
WHERE manager_id IN (SELECT manager_id
                     FROM employees
                     WHERE employee_id IN (174, 141))
AND department_id IN (SELECT department_id
                     FROM employees
                     WHERE employee_id IN (174, 141))
AND employee_id NOT IN(174, 141);
```

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary < ANY (SELECT salary
                   FROM employees
                   WHERE job_id = 'IT_PROG')
AND job_id <> 'IT_PROG';
```

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary < ALL (SELECT salary
                     FROM employees
                     WHERE job_id = 'IT_PROG')
AND job_id <> 'IT_PROG';
```

- 다중컬럼 서브쿼리

```
SELECT employee_id, first_name, department_id, salary
FROM employees
WHERE (department_id, salary) IN (SELECT department_id, min(salary)
                                FROM employees
                                GROUP BY department_id)

ORDER BY department_id;
```

- 결과가 나오지 않는 원인은?

```
SELECT last_name
FROM employees
WHERE employee_id NOT IN (SELECT manager_id
                          FROM employees);
```

<연습문제>

1. employees 테이블에서 Abel과 동일한 부서에 소속된 직원들의 last_name과 hire_date를 출력하되 비교의 대상인 Abel은 제외하시오.
2. employees 테이블에서 평균 이상의 급여를 받는 직원들의 employee_id, last_name, salary를 출력하되 급여를 기준으로 오름차순 하시오.
3. employees 테이블에서 last_name에 'u'가 포함된 직원과 같은 부서에 근무하는 모든 직원의 employee_id, last_name을 출력하시오.
4. employees 테이블과 departments 테이블을 사용하여 구문을 작성하시오.
location_id가 1700인 부서에 소속된 직원들의 employee_id, last_name, department_id, job_id를 출력하시오.
5. employees 테이블에서 평균 이상의 급여를 받으면서 last_name에 'u'가 포함된 직원과 동일한 부서에 소속된 직원들의 employee_id, last_name, salary를 출력하시오.
6. employees 테이블에서 본인이 매니저의 역할을 하는 직원들의 employee_id, last_name을 출력하시오.
7. employees 테이블과 departments 테이블을 사용하여 구문을 작성하시오.
직원이 소속되어 있지 않은 빈 부서의 department_id, department_name을 출력하시오.

8. 데이터조작어(DML)

1. 데이터조작어(DML)

- 테이블에 새로운 행 추가(insert)
- 테이블의 기존 행 수정(update)
- 테이블의 기존 행 삭제(delete)

2. 데이터 삽입(insert)

[문법]

```
INSERT INTO table [(column [, column...])]  
VALUES (value [, value...]);
```

[예제]

```
INSERT INTO departments(department_id,  
                        department_name, manager_id, location_id)  
VALUES (70, 'Public Relations', 100, 1700);
```

1 rows inserted

[null값 삽입하는 방법]

```
INSERT INTO departments (department_id,  
                        department_name)  
VALUES (30, 'Purchasing');
```

1 rows inserted

```
INSERT INTO departments  
VALUES (100, 'Finance', NULL, NULL);
```

1 rows inserted

[subquery가 사용된 insert 구문]

- sales_reps 테이블 생성 후 실습 가능!

```
INSERT INTO sales_reps(id, name, salary, commission_pct)  
SELECT employee_id, last_name, salary, commission_pct  
FROM employees  
WHERE job_id LIKE '%REP%';
```

4 rows inserted

MySQL

- employees 테이블의 복사본으로 copy_emp 테이블을 생성하고 데이터를 삽입하시오.

```
// employees 테이블과 동일한 구조의 copy_emp 테이블 생성하기
create table copy_emp
as select *
  from employees
 where 1 = 2;

desc copy_emp

select *
from copy_emp;

// copy_emp 테이블에 employees 테이블에 있는 107명의 사원 정보 동일하게
  삽입하기
insert into copy_emp
  select *
  from employees;
```

3. 데이터 수정(update)

[문법]

```
UPDATE      table
SET         column = value [, column = value, ...]
[WHERE      condition];
```

[예제]

```
UPDATE employees
SET    department_id = 50
WHERE  employee_id = 113;
```

1 rows updated

[update 구문 작성 시 where절을 생략한 경우]

- MySQL Workbench - [Query] - [Auto-Commit Transactions] 해제 후 작업하기

```
UPDATE    copy_emp
SET       department_id = 110;
```

22 rows updated

MySQL

[subquery가 사용된 update 구문]

```
update copy_emp
set job_id = ( select job_id
               from employees
               where employee_id = 205 ),
    salary = ( select salary
               from employees
               where employee_id = 205 )
where employee_id = 113;
```

```
update copy_emp
set salary = salary * 1.1
where department_id = ( select department_id
                        from departments
                        where location_id = 1800 );
```

4. 데이터 삭제(delete)

[문법]

```
DELETE [FROM]    table
[WHERE          condition];
```

[예제]

```
DELETE FROM departments
WHERE department_name = 'Finance';
```

1 rows deleted

[delete 구문 작성 시 where절을 생략한 경우]

- MySQL Workbench - [Query] - [Auto-Commit Transactions] 해제 후 작업하기

```
DELETE FROM copy_emp;
```

22 rows deleted

[subquery가 사용된 delete 구문]

```
delete from copy_emp
where department_id = ( select department_id
                        from departments
                        where location_id = 1800);
```

5. 트랜잭션제어어(TCL)

- 트랜잭션이란? 하나의 논리적인 작업 단위로 여러 개의 DML이 하나의 트랜잭션을 구성할 수 있음.
- 트랜잭션 제어 명령어 :

commit	변경 작업을 영구히 저장하는 명령어 SQL> commit;
rollback	변경 작업을 트랜잭션 처음으로 되돌리는 명령어 SQL> rollback;
savepoint	트랜잭션 진행 중 되돌아갈 지점을 생성하는 명령어 [예시] insert ---; update ---; savepoint 포인트명; delete ---; update ---; rollback to 포인트명;

- MySQL Workbench에서 트랜잭션 설정하기 → [Query] 메뉴
 - ① [Auto-Commit Transactions]
 - 체크 설정 : 자동 커밋 기능 설정, 즉 DML 실행 시 바로 저장됨.
알아서 자동 커밋되므로 편하나 작업 실수 시 되돌릴 수 없음.
 - 체크 해제 : 수동 커밋, 수동 롤백 가능함, 즉 DML 실행 후 커밋, 롤백 결정해야함.
작업 후 수동으로 커밋, 롤백을 실행하여 작업을 마무리를 해줘야하나 작업 실수 시 되돌릴 수 있음.
 - ② [Commit Transaction] : 수동으로 커밋하는 메뉴
 - ③ [Rollback Transaction] : 수동으로 롤백하는 메뉴 SQL> rollback;

MySQL

<연습문제>

- MySQL Workbench - [Query] - [Auto-Commit Transactions] 해제 후 작업하기

※ MY_EMPLOYEE 테이블에 데이터를 삽입합니다.

- MY_EMPLOYEE라는 테이블을 생성합니다.

```
CREATE TABLE my_employee  
(id int PRIMARY Key,  
last_name VARCHAR(25),  
first_name VARCHAR(25),  
userid VARCHAR(8),  
salary int);
```

- 열 이름을 식별하도록 MY_EMPLOYEE 테이블의 구조를 기술합니다.

```
DESC my_employee;
```

- 다음 데이터를 MY_EMPLOYEE 테이블에 추가하는 INSERT문을 작성하시오.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
1	Patel	Ralph	rpatel	895
2	DanCS	Betty	bdanCS	860
3	Biri	Ben	bbiri	1100
4	Newman	Chad	cnewman	750

- 테이블에 추가한 내용을 확인하시오.
- 테이블에 삽입한 내용을 영구히 저장하시오.

※ MY_EMPLOYEE 테이블에서 데이터를 갱신하고 삭제합니다.

- ID가 3인 사원의 last_name을 Drexler로 변경하시오.
- 급여가 \$900 미만인 모든 사원에 대해 급여를 \$1000로 변경하시오.

MySQL

8. 테이블에 변경 작업한 내용을 확인하시오.

9. MY_EMPLOYEE 테이블에서 Betty Dancs란 사원을 삭제하시오.

10. 테이블에 변경 작업한 내용을 확인하시오.

11. 보류 중인 모든 변경 사항을 커밋하시오.

※ MY_EMPLOYEE 테이블에 대한 데이터 트랜잭션을 제어합니다.

12. 다음 데이터를 MY_EMPLOYEE 테이블에 추가하는 INSERT문을 작성하시오.

ID	LAST_NAME	FIRST_NAME	USERID	SALARY
5	Ropeburn	Audrey	aropebur	1550

13. 테이블에 추가한 내용을 확인하시오.

14. insert 작업까지 진행한 현재 위치에 저장점(savepoint)을 생성하시오.

15. MY_EMPLOYEE 테이블에서 모든 행을 삭제하시오.

16. 테이블이 비어 있는지 확인하시오.

17. 이전의 INSERT 작업은 삭제하지 않은 채로 최근의 DELETE 작업만 취소하시오.

18. INSERT 작업은 삭제하지 않은 채로 최근의 DELETE 작업만 취소되었는지 확인하시오.

19. 작업한 내용을 영구히 저장하시오.

9. 데이터정의어(DDL) - Table

※ shopdb 데이터베이스(스키마)에서 테이블 생성 작업 진행 예정
테이블 목록 확인 후 불필요한 테이블은 삭제하고 시작하기!

```
use shopdb;  
show tables;  
drop table test1;
```

1. 테이블 생성(create table)

- 테이블 생성 시 테이블명, 컬럼명, 데이터타입 지정해야함.
- 옵션 : Default값, 제약조건
- default값이 포함된 테이블 생성 예제

```
create table dept  
(deptno int,  
  dname varchar(14),  
  loc varchar(13),  
  create_date datetime default now( ));
```

2. 제약조건

1) not null 제약조건

- 컬럼에 null값을 허용하지 않는 제약조건
- 값이 반드시 있어야하는 필수 컬럼에 not null 제약조건을 지정한다.
- null값 저장 시 고정 길이 문자형(char)은 공간을 모두 차지하지만, 가변 길이 문자형(varchar)은 공간을 차지하지 않으므로 null값이 많이 입력되는 컬럼은 가변 길이 문자형(varchar)을 사용하는 것을 권장한다.
- not null 제약조건이 포함된 테이블 생성 예제

```
create table test1  
( id int not null,  
  name varchar(30) not null,  
  jumin varchar(13) not null,  
  job varchar(20),  
  email varchar(20),  
  phone varchar(20) not null,  
  start_date date );  
  
desc test1;
```

MySQL

2) unique 제약조건

- 중복되지 않는 유일한(고유한) 값을 입력해야 하는 제약조건
- 중복된 값은 허용하지 않으나 null값은 허용함. 또한 null값은 모르는 값으로 null값끼리는 동일한 값이 아니므로 여러 개 입력될 수 있다.
- 중복된 값이 들어와서는 안되는 주민번호, 전화번호, 메일 등의 컬럼에 활용된다.
- unique 제약조건이 포함된 테이블 생성 예제

```
create table test2
( id int not null unique,
  name varchar(30) not null,
  jumin varchar(13) not null unique,
  job varchar(20),
  email varchar(20) unique,
  phone varchar(20) not null unique,
  start_date date );

desc test2;
```

3) primary key 제약조건

- 테이블의 행을 고유하게 식별해 줄 수 있는 컬럼(대표 컬럼)에 선언하는 기본키 제약조건
- not null + unique의 성격을 모두 가지는 제약조건
- 주로 사번, 학번, 회원번호, 제품번호, 주문번호 등의 컬럼에 선언한다.
- primary key 제약조건이 포함된 테이블 생성 예제

```
create table test3
( id int primary key,
  name varchar(30) not null,
  jumin varchar(13) not null unique,
  job varchar(20),
  email varchar(20) unique,
  phone varchar(20) not null unique,
  start_date date );

desc test3;
```

4) foreign key 제약조건

- 두 테이블 사이의 관계를 선언함으로써 데이터의 무결성을 보장해 주는 외래키 제약조건
- 자기 자신 테이블이나 다른 테이블의 특정 컬럼을 참조하는 제약조건
- 단, primary key 또는 unique 제약조건이 선언된 컬럼만 참조할 수 있음!!!
- foreign key 제약조건이 선언된 컬럼 = 자식 컬럼

MySQL

- foreign key 제약조건이 참조하는 컬럼 = 부모 컬럼
- 자식 컬럼에는 부모 컬럼에 있는 데이터 중 하나만 삽입, 수정될 수 있다.
- foreign key 제약조건이 포함된 테이블 생성 예제

```
create table test4  
( t_num int primary key,  
  t_id int,  
  title varchar(20) not null,  
  story varchar(100) not null,  
  foreign key(t_id) references test3(id) );  
  
desc test4;
```

5) check 제약조건

- 해당 컬럼이 만족해야하는 조건을 지정하는 제약조건
- check 제약조건이 포함된 테이블 생성 예제

```
create table test5  
( id int(10) primary key,  
  name varchar(30) not null,  
  jumin varchar(13) not null unique check (length(jumin)=13),  
  job varchar(20),  
  email varchar(20) unique,  
  phone varchar(20) not null unique,  
  start_date date check (start_date >= '2005-01-01') );  
  
desc test5;
```

6) 테이블에 선언된 제약조건 정보 조회

```
show databases;

use information_schema;

// test5 테이블에 선언된 제약조건 정보 조회
select *
from table_constraints
where table_name = 'test5';

// shopdb 데이터베이스(스키마)에 존재하는 테이블에 선언된
  모든 제약조건 정보 조회
select *
from table_constraints
where constraint_schema = 'shopdb';
```

3. 서브쿼리를 사용한 테이블 생성

```
create table dept80
as select employee_id, last_name, salary*12 as annsal, hire_date
   from employees
   where department_id = 80;

desc dept80;

select *
from dept80;
```

- 서브쿼리를 사용해서 테이블 생성 시 원본 테이블의 구조 및 데이터 모두 복사된 테이블이 생성됨.
- 단, 제약조건은 not null 제약조건만 복사됨.
- 테이블 백업 또는 테스트용 테이블 생성 시 주로 활용함.

4. 테이블 수정(alter table)

1) 컬럼 추가

- 테이블 생성 후 컬럼 추가 시 기본적으로 마지막 컬럼으로 추가됨.
- 초기 값은 null값이 삽입되어 있음.

```
alter table dept80
add job_id varchar(9);
```


MySQL

- default값 지정 시에는 초기 값으로 default값이 삽입되어 있음.

```
alter table dept80
add email varchar(30) default '미입력';
```

- 가장 앞에 또는 특정 컬럼 뒤에 컬럼 추가하는 방법

```
alter table dept80
add emp_number int first;

alter table dept80
add salary int default 300 not null after last_name;
```

2) 컬럼 수정

- 데이터타입, 컬럼 사이즈, default값 변경

```
alter table dept80
modify salary bigint;

alter table dept80
modify last_name varchar(30) not null;

alter table dept80
modify salary bigint default 500 not null;
=> 저장된 default값이 변경되는게 아니라 변경 이후 데이터 삽입 시부터 적용됨.
```

- 컬럼명 변경

```
alter table dept80
rename column hire_date to start_date;
```

3) 컬럼 삭제

- foreign key 제약조건이 참조하는 부모 컬럼인 경우에는 제약조건을 먼저 삭제해야함.

```
alter table dept80
drop emp_number;
```

MySQL

4) 제약조건 추가

```
// primary key와 unique 제약조건 추가 문법 동일함.  
alter table dept80  
add primary key(employee_id);  
  
// not null 제약조건 추가 문법  
alter table dept80  
modify annsal double(22,0) not null;  
  
// check 제약조건 추가 문법  
alter table dept80  
add check (salary > 100);  
  
// foreign key 제약조건 추가 문법  
alter table dept80  
add mgr_id int default 150;  
  
desc dept80;  
  
select *  
from dept80;  
  
alter table dept80  
add foreign key(mgr_id) references dept80(employee_id);  
  
desc dept80;
```

5) 제약조건 삭제

- 제약조건 정보 조회

```
use information_schema;  
  
select *  
from table_constraints  
where table_name = 'dept80';
```

```
// primary key 제약조건 삭제 시 오류 발생됨.
// foreign key 제약조건이 선언된 컬럼이 참조하고 있는 부모 컬럼에 선언된
  primary key 또는 unique 제약조건은 삭제할 수 없음.
// 삭제를 원하는 경우 foreign key 제약조건을 먼저 삭제해야함.
alter table dept80
drop primary key;

// foreign key 제약조건 삭제 문법
alter table dept80
drop foreign key dept80_ibfk_1;

// check 제약조건 삭제 문법
alter table 테이블명
drop check check제약조건명;

// unique 제약조건 삭제 문법
alter table 테이블명
drop index unique제약조건명;

// not null 제약조건 삭제 문법
alter table dept80
modify annsal double(22,0) null;
```

5. 테이블 삭제(drop table)

- 데이터베이스로부터 테이블을 삭제함.
- 테이블 구조, 데이터, 제약조건 등 모두 삭제됨.

```
drop table dept80;
```

6. 테이블 절단(truncate table)

- 테이블로부터 모든 행을 삭제함.

```
truncate table dept80;
```

<연습문제>

※ shopdb 데이터베이스(스키마)에서 테이블 생성 연습문제 작업하기!

1. 테이블 이름 : **TITLE**

Column Name	TITLE_ID	TITLE	DESCRIPTION	RATING	CATEGORY	RELEASE_DATE
Key Type	PK					
Null/Unique		NN	NN			
Check				G, PG, R, NC17, NR	DRAMA, COMEDY, ACTION, CHILD, SCIFI, DOCUMENTARY	
Data type	INT	VARCHAR	VARCHAR	VARCHAR	VARCHAR	DATE
Length	-	60	400	4	20	

2. 테이블 이름 : **TITLE_COPY**

Column Name	COPY_ID	TITLE_ID	STATUS
Key Type	PK	PK, FK	
Null/Unique			NN
Check			AVAILABLE, DESTROYED, RENTED, RESERVED
FK Table		TITLE	
FK Column		TITLE_ID	
Data type	INT	INT	VARCHAR
Length	-	-	15

10. 데이터정의어(DDL) - View

1. View란?

- 개념 : 하나이상의 테이블을 기반으로 생성은 되었으나 물리적으로 존재하지는 않고 DB에 정의만 되어 있는 가상의 논리적인 테이블
- 목적 : ① 보안에 도움이 된다.
② 복잡한 쿼리 구문을 단순화 시킬 수 있다.
- 사용방법 : 테이블과 동일하게 사용된다.

2. View 생성(create view)

```
create view empvu80
as select employee_id, last_name, salary, department_id
   from employees
   where department_id = 80;

desc empvu80;

select *
from empvu80;
```

```
create view deptvu
as select *
   from departments
   where department_id > 200;

desc deptvu;

select *
from deptvu;

insert into deptvu
values (280, 'AAA', null, 1700);

select *
from deptvu;

select *
from departments;
```

```
// 보안성 예제
use shopdb;

create view member_vu
as select member_id, member_name, birth, job
   from members;

desc member_vu;

select *
from member_vu;
```

```
// 단순성 예제
use hr;

create view dept_sal_vu
as select d.department_name, sum(e.salary) as "급여 합계",
       avg(e.salary) as "급여 평균", min(e.salary) as "최소 급여",
       max(e.salary) as "최대 급여"
   from employees e join departments d
   on e.department_id = d.department_id
  group by d.department_name
 order by d.department_name;

desc dept_sal_vu;

select *
from dept_sal_vu;
```

MySQL

3. View 수정

```
alter view empvu80
as select employee_id, last_name, salary, email, department_id
   from employees
   where department_id = 80;

desc empvu80;

select *
from empvu80;
```

4. View 삭제(drop view)

```
drop view empvu80;
```

5. View 정보 조회

```
use information_schema;

show tables;

select *
from views
where table_schema = 'hr';
```