

Day 23

☰ Tags	SQL
📅 Date	@2022년 8월 1일
▼ 강의 번호	AUS 101

-SQL-

(강사님 공유해주신 교재로 수업)

-. 왜 배우는가? : 웹개발 하려고

-. 구문은 대문자로 하지만, 소문자로 해도 상관은 없다.

```
mysql< SELECT name
      -> FROM corporation
      -> WHERE corp_id = 27;
+-----+
| name                |
+-----+
| Acme Paper Corporation |
+-----+
```

p.15 - 예제 쉽게 다운받는 방법 - My SQL

-. 사회 나가면 오라클이 사실 대부분.

< 자료형 >

1) 문자 데이터 (p.18)

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

char - 고정길이 : 8글자 데이터를 입력할 때 나머지 글자는 모두 공백으로 채워서 저장

varchar - 가변길이 : 8글자 데이터를 입력할 때 8글자만 저장된다. → 근데 애만 (죽어라)
쓴다. 주로 사용하는 문자 자료형

2) 텍스트 데이터 (p.20)

: 긴 문자열을 저장할 때 사용한다.

text : 주로 사용하는 텍스트 자료형

tinytext ← 잘 안쓴다.

3) 숫자 데이터 (p.21)

int 주로 사용하는 숫자 자료형

tinyint

smallint

bigint

4) 날짜 데이터 (p.23)

timestamp - 현재 날짜와 시간을 자동입력 (할 때 많이 쓴다.) → 실제 많이 쓰는 것. : 직접 입력 (주로 회원가입을 통한)

datetime - 날짜와 시간 : 자동 입력

< 테이블 작성 > (p.25)

: 가장 중요하다.

1) 설계 (디자인)

: 테이블에 저장할 적당한 항목들과 그 항목들을 저장할 데이터 형과 크기를 설계

이름, 주소, 전화번호, 성, 음식...

2) 정제

: 테이블 작성시에는 기본키 설정이 중요! - ' PRIMARY KEY '

: 이름의 경우에는 성과 이름으로 분리 (를 하면 데이터 분석 시 이점을 가질 수 있다.)

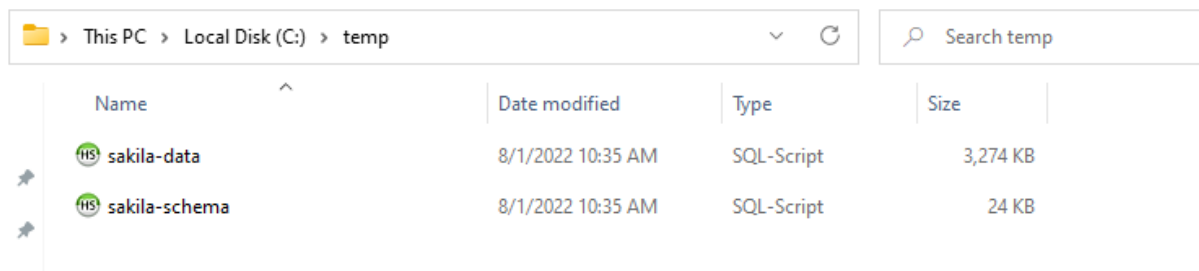
주소의 경우에도 하나의 필드로 저장하는 것보다 예를 들면 시, 군, 구 별로 따로 분리

3) SQL 구문 생성

```
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)
);
```

+) <https://dev.mysql.com/doc/index-other.html> 에서 sakila 예제 파일 다운 → ‘ sakila-db.zip ‘ 파일로.

→ 파일 압축 푼 것을 아래와 같은 경로로 만들어 붙여넣기



Name	Date modified	Type	Size
sakila-data	8/1/2022 10:35 AM	SQL-Script	3,274 KB
sakila-schema	8/1/2022 10:35 AM	SQL-Script	24 KB

→ maria client cmd로 가서

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

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

각 실행.

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

< 테이블 수정 >

1) 데이터 삽입

- 데이터 추가할 테이블 이름
- 데이터 추가할 테이블의 열 이름
- 열에 넣을 값

테이블에 데이터 입력

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

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

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

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

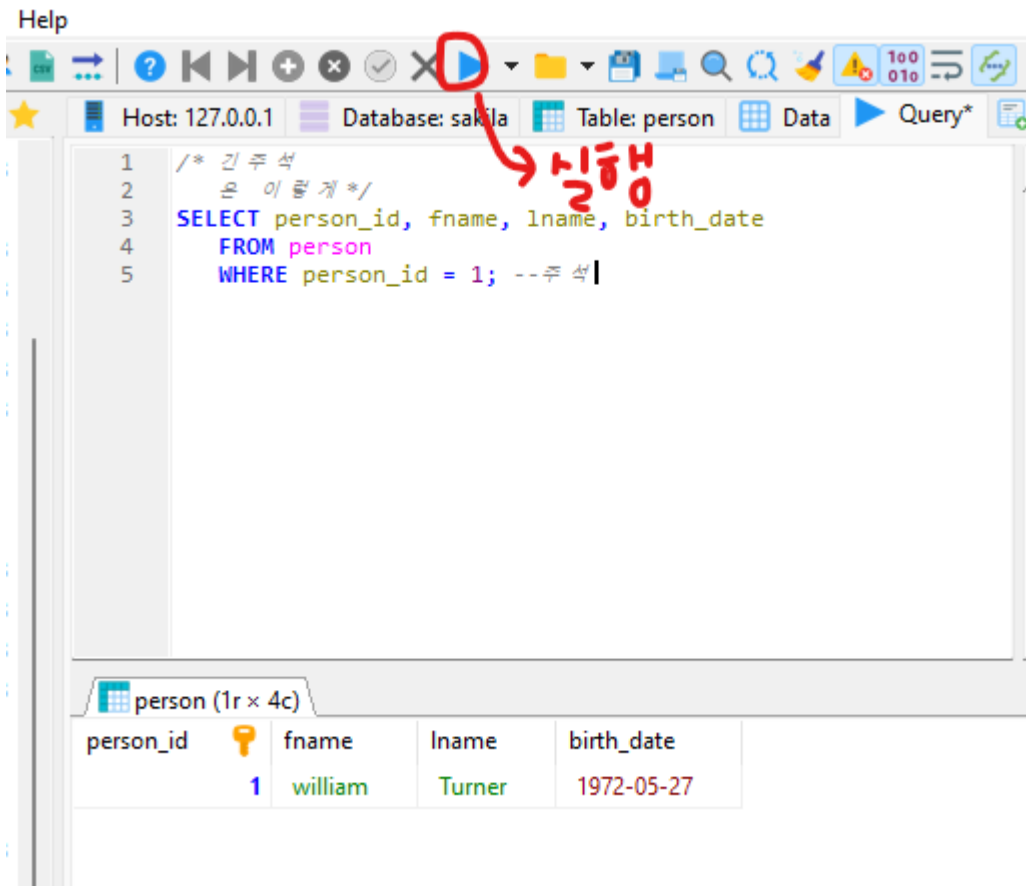


```
SELECT person_id, fname, lname, birth_date
FROM person
WHERE person_id = 1;
```

SQL 주석 2가지



```
/* 긴주석
   은 이렇게*/
SELECT person_id, fname, lname, birth_date
FROM person
WHERE person_id = 1; -- 한줄 주석
```



```

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_food이라는 테이블로부터 조건은 person_id = 1인 값만 순서를 food열을 오름차순(기본) 정렬하여 조회



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

Host: 127.0.0.1	Database: sakila	Table: favorite_foo
-----------------	------------------	---------------------

1	SELECT food
2	FROM favorite_food
3	WHERE person_id = 1
4	ORDER BY food;

favorite_food (3r × 1c)	
food	
cookies	
nachos	
pizza	

```
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');
```



```
SELECT *
FROM person;
```

The screenshot shows a MySQL query client window. The top toolbar contains various icons for file operations, execution, and search. Below the toolbar, the status bar indicates 'Host: 127.0.0.1', 'Database: sakila', 'Table: person', 'Data', and 'Query*'. The query editor shows the following SQL statement:

```
1 SELECT *
2 FROM person
```

Below the query editor, the results are displayed in a table format. The table has 10 columns: person_id, fname, lname, gender, birth_date, street, city, state, country, and postal_code. The results show three rows of data:

person_id	fname	lname	gender	birth_date	street	city	state	country	postal_code
0	julia	Kim	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)
1	william	Turner	(NULL)	1972-05-27	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)
2	Susan	Smith	F	1975-11-02	23 Maple St.	Arlington	VA	USA	20220

입력 후 잘 들어갔나 확인해 봄.

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

데이터 업데이트



```
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;
```

```
DELETE FROM person
WHERE person_id = 2;
```

person_id	fname	lname	gender	birth_date	street	city	state
0	julia	Kim	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)
1	william	Turner	(NULL)	1972-05-27	1225 Tremont St.	Boston	MA
2	Susan	Smith	F	1975-11-02	23 Maple St.	Arlington	VA

person_id	fname	lname	gender	birth_date	street	city	state
0	julia	Kim	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)
1	william	Turner	(NULL)	1972-05-27	1225 Tremont St.	Boston	MA

수잔 삭제됨.

에러)

1. 고유 키 값이 중복된 데이터를 입력하려고 시도할 때 에러 발생


```
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'
```

1. 존재하지 않는 외래 키 foreign key를 참조할 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'))
```

1. 열 값 위반. 선언한 데이터형을 벗어났을 때 에러 발생

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

1. 잘못된 날짜 변환. (날짜의 기본형인 년 - 월 - 일로 입력되지 않고 월 - 일 - 년으로 입력되어 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
```

< 테이블 제거 >

: DROP TABLE 테이블이름;

```
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)
```

SOURCE C:/temp/LearningSQLExample.sql;

찾는게 없을 때,



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

있을 때,



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

SELECT - 쿼리 결과에 포함 시킬 열들 결정

FROM - 결과를 검색할 테이블, 테이블들을 조인하는 방법 등

WHERE - 원하지 않는 데이터를 걸러내는 조건 설정

GROUP BY - 공통열 값을 기준으로 행들을 그룹화

HAVING - 원하지 않는 그룹을 걸러내는 조건 설정

ORDER BY - 하나 또는 하나 이상의 열들을 기준으로 최종 결과의 행들을 정렬

1. SELECT

select 절을 완전하게 이해하려면 from 절을 먼저 이해해야 한다.



```
SELECT * FROM department;
```

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

또는 하나의 열만 선택하여 결과를 볼 수도 있다.

숫자나 문자를 그냥 출력

기존열의 값을 계산한 결과를 출력

함수를 사용한 결과



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

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

→ AS는 생략가능하다.

Host: 127.0.0.1	Database: bank	Query*
-----------------	----------------	--------


```

1  SELECT emp_id,
2     'ACTIVE' AS 상태,
3     emp_id * 3.14159 AS empid_x_pi,
4     UPPER(lname) AS last_name_upper
5  FROM employee;
6

```


employee (18r x 4c)			
emp_id	상태	empid_x_pi	last_name_upper
1	ACTIVE	3.14159	SMITH
2	ACTIVE	6.28318	BARKER
3	ACTIVE	9.42477	TYLER
4	ACTIVE	12.56636	HAWTHORNE
5	ACTIVE	15.70795	GOODING
6	ACTIVE	18.84954	FLEMING
7	ACTIVE	21.99113	TUCKER
8	ACTIVE	25.13272	PARKER

< 중복 제거 DISTINCT >

: 상황에 따라 쿼리가 중복된 행을 반환할 수 있다. 고유한 하나의 값만 남기고 나머지는 제거한 값을 확인 할 수 있다. (실제 제거한 것이 아닌, 보여주지만 제거한 것을 보여주는 것. 실제로 제거되지 않고 데이터가 존재 함. 주의하자!)

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

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

< FROM 절 >

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

: FROM 절은 쿼리에 사용되는 테이블을 명시할 뿐만 아니라 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

< 파생 테이블 (← 임시 테이블) >

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

Virtual Table → 가상 테이블

```
mysql> CREATE VIEW employee_vw AS  
-> SELECT emp_id, fname, lname,  
->        YEAR(start_date) start_year  
-> FROM employee;  
Query OK, 0 rows affected (0.10 sec)
```

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

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

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

p.51 테이블 연결

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

```
SELECT 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 e INNER JOIN department d
ON e.dept_id = d.dept_id;
```

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

```
SELECT 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 53. WHERE 절

: where 절은 결과에 출력되기를 원하지 않는 행을 걸러내는 방법이다.

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

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

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

Host: 127.0.0.1	Database: bank	Table: employee	Data	Query*
1	SELECT	emp_id, fname, lname, start_date, title		
2	FROM	employee		
3	WHERE	title = 'Head Teller';		

employee (4r x 5c)				
emp_id	fname	lname	start_date	title
6	Helen	Fleming	2004-03-17	Head Teller
10	Paula	Roberts	2002-07-27	Head Teller
13	John	Blake	2000-05-11	Head Teller
16	Theresa	Markham	2001-03-15	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';
```

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

< Group by 절과 Having 절 >

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

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

< ORDER BY 절 >

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

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

open_emp_id	product_cd
10	CHK
10	SAV
10	CD
10	CHK
10	SAV
13	CHK
13	MM
1	CHK
1	SAV
1	MM
16	CHK
1	CHK
1	CD

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

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

정렬의 기준이 여러개인 경우는 첫번째 정렬을 마친 값들 중 동일한 값들만 다시한번 정렬

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

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

정렬의 기본은 오름차순으로 적시 하지 않으면 오름차순 정렬되고

DESC 를 적으면 내림차순으로 정렬된다.

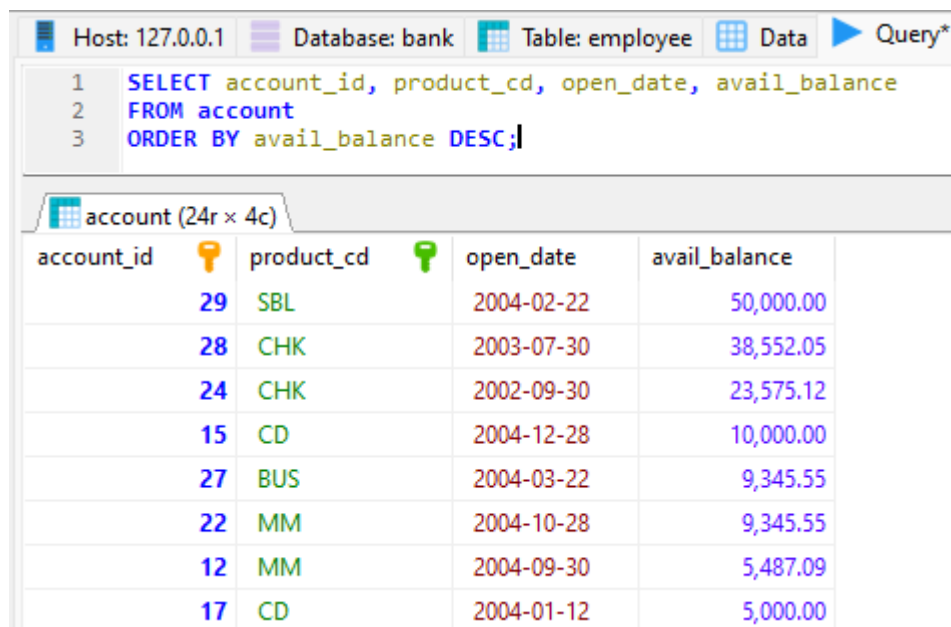
```
mysql> SELECT account_id, product_cd, open_date, avail_balance  
-> FROM account  
-> ORDER BY avail_balance DESC;
```

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

SELECT account_id, product_cd, open_date, avail_balance

FROM account

ORDER BY avail_balance DESC;



account_id	product_cd	open_date	avail_balance
29	SBL	2004-02-22	50,000.00
28	CHK	2003-07-30	38,552.05
24	CHK	2002-09-30	23,575.12
15	CD	2004-12-28	10,000.00
27	BUS	2004-03-22	9,345.55
22	MM	2004-10-28	9,345.55
12	MM	2004-09-30	5,487.09
17	CD	2004-01-12	5,000.00

```
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);

```

사원 테이블에서 사원 번호와 이름과 월급을 출력해 보겠습니다.

Host: 127.0.0.1 Database: bank Table: emp Data Query*

1 SELECT EMPNO, ENAME, SAL

2 FROM emp;

emp (14r x 3c)

EMPNO	ENAME	SAL
7,839	KING	5,000
7,698	BLAKE	2,850
7,782	CLARK	2,450
7,566	JONES	2,975
7,654	MARTIN	1,250
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
7,788	SCOTT	3,000
7,876	ADAMS	1,100
7,934	MILLER	1,300

Host: 127.0.0.1	Database: bank	Table: emp	Data	Query*
1	SELECT empno, ename, sal			
2	FROM emp;			
3				
4	SELECT empno, ename, sal FROM emp;			
5				
6	SELECT empno, ename, sal			
7	FROM emp;			
8				

emp (14r x 3c)		
empno	ename	sal
7,839	KING	5,000
7,698	BLAKE	2,850
7,782	CLARK	2,450
7,566	JONES	2,975
7,654	MARTIN	1,250
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
7,788	SCOTT	3,000
7,876	ADAMS	1,100
7,934	MILLER	1,300

이름의 대문자, 소문자,, 줄바꿈 안하고,, 들여쓰기를 많이 해도 상관없이 다 읽어준다.

사원 테이블을 모든 열(column)들을 전부 출력해 보겠습니다.

Host: 127.0.0.1	Database: bank	Table: emp	Data	Query*	Filter ...
1	SELECT empno, ename, job, mgr, hiredate, sal, comm, deptno				> Columns in emp
2	FROM emp;				> SQL functions
3					> SQL keywords
4	SELECT *				> Snippets
5	FROM emp;				
6					

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
7,788	SCOTT	ANALYST	7,566	1982-12-22	3,000	(NULL)	20
7,876	ADAMS	CLERK	7,788	1983-01-15	1,100	(NULL)	20
7,934	MILLER	CLERK	7,782	1982-01-11	1,300	(NULL)	10

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

출력 결과

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

Host: 127.0.0.1
Database: bank
Table: emp
Data
Query*

```

1 SELECT empno AS '사원 번호',
2        ename AS 이름,
3        sal AS '사원 Salary'
4 FROM emp;
5

```

emp (14r x 3c)

사원 번호	이름	사원 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
7,788	SCOTT	3,000
7,876	ADAMS	1,100
7,934	MILLER	1,300

사원 테이블의 이름과 월급을 서로 붙여서 출력해 보겠습니다.

KING5000
BLAKE2850
CLARK2450
JONES2975
⋮

Host: 127.0.0.1
Database: bank
Table: emp
Data
Query*

```

1  /* 오라클용
2  SELECT ename || sal
3  FROM emp; */
4
5  SELECT CONCAT(ename, sal)
6  FROM emp;
7

```

emp (14r × 1c)

CONCAT(ename,sal)
KING5000
BLAKE2850
CLARK2450
JONES2975
MARTIN1250
ALLEN1600
TURNER1500
JAMES950
WARD1250
FORD3000
SMITH800
SCOTT3000
ADAMS1100
MILLER1300

[오라클에서만 동작] ⇒ SELECT ename || sal FROM emp ;

[오라클에서만 동작]

```

SELECT ename || sal
FROM emp;

```

직업정보

KING 의 직업은 PRESIDENT 입니다

BLAKE 의 직업은 MANAGER 입니다

CLARK 의 직업은 MANAGER 입니다

JONES 의 직업은 MANAGER 입니다

MARTIN 의 직업은 SALESMAN 입니다

ALLEN 의 직업은 SALESMAN 입니다

TURNER 의 직업은 SALESMAN 입니다

JAMES 의 직업은 CLERK 입니다

Host: 127.0.0.1	Database: bank	Table: emp	Data	Query*
1	SELECT CONCAT(ename, '의 직업은 ', job, '입니다.')			
2	FROM emp;			
3				
emp (14r x 1c)				
CONCAT(ename, '의 직업은 ', job, '입니다.)				
KING의 직업은 PRESIDENT입니다.				
BLAKE의 직업은 MANAGER입니다.				
CLARK의 직업은 MANAGER입니다.				
JONES의 직업은 MANAGER입니다.				
MARTIN의 직업은 SALESMAN입니다.				
ALLEN의 직업은 SALESMAN입니다.				
TURNER의 직업은 SALESMAN입니다.				
JAMES의 직업은 CLERK입니다.				
WARD의 직업은 SALESMAN입니다.				
FORD의 직업은 ANALYST입니다.				
SMITH의 직업은 CLERK입니다.				
SCOTT의 직업은 ANALYST입니다.				
ADAMS의 직업은 CLERK입니다.				
MILLER의 직업은 CLERK입니다.				

사원 테이블에서 직업을 출력하는데 중복된 데이터를 제외하고 출력해 보겠습니다.

JOB
SALESMAN
CLERK
ANALYST
MANAGER
PRESIDENT

Host: 127.0.0.1 Database: bank Table: emp Data Query*

```
1  /*
2  SELECT DISTINCT job
3    FROM emp;
4  */
5
6  SELECT UNIQUE job
7    FROM emp;
8
```

emp (5r x 1c)

job
PRESIDENT
MANAGER
SALESMAN
CLERK
ANALYST

중복 제거 - DISTINCT 외에 UNIQUE 가 있다.

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

Host: 127.0.0.1
Database: bank
Table: emp
Data
Query*

```

1 SELECT ename, sal
2 FROM emp
3 ORDER BY sal;
4

```

emp (14r x 2c)

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

오름차순 (작은 순서대로)

Host: 127.0.0.1	Database: bank	Table: emp	Data	Query*
1	SELECT	ename, sal		
2	FROM	emp		
3	ORDER BY	sal DESC;		
4				

emp (14r × 2c)	
ename	sal
KING	5,000
SCOTT	3,000
FORD	3,000
JONES	2,975
BLAKE	2,850
CLARK	2,450
ALLEN	1,600
TURNER	1,500
MILLER	1,300
WARD	1,250
MARTIN	1,250
ADAMS	1,100
JAMES	950
SMITH	800

내림차순 (큰 순서대로)

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

Host: 127.0.0.1 Database: bank Table: emp Data Query*

```
1 SELECT ename, deptno, sal
2 FROM emp
3 ORDER BY deptno ASC, sal DESC;
4
```

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
WARD	30	1,250
MARTIN	30	1,250
JAMES	30	950

기타)

영어 - EBS 강추 (오디오 어학당 - 입트영, 귀트영, 파워00 → 라디오)