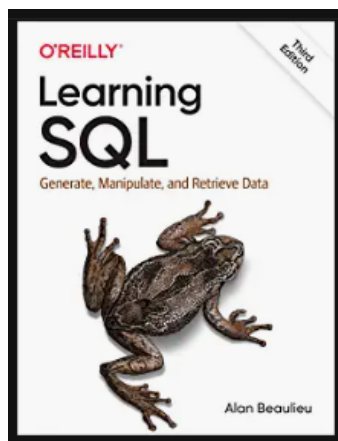


Day23; 20220801(DB)

🕒 작성일시	@2022년 8월 1일 오전 9:31
▼ 강의 번호	lesson 23
📅 유형	@2022년 8월 1일
📎 자료	
☑ 복습	<input type="checkbox"/>
☰ 속성	

🔗 [https://github.com/u8yes/DATABASELAB/blob/main/Day23_20220801\(DB\).pdf](https://github.com/u8yes/DATABASELAB/blob/main/Day23_20220801(DB).pdf)

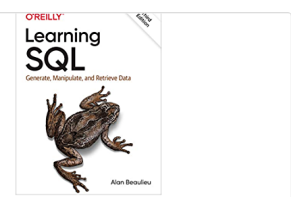


Learning SQL: Generate, Manipulate, and Retrieve Data

Amazon.com: Learning SQL: Generate, Manipulate, and Retrieve Data: 9781492057611:

Beaulieu, Alan: Books

[a https://www.amazon.com/-/ko/dp/1492057614/ref=sr_1_1?crid=80HRM2JMHD9&keywords=Learning+SQL&qid=1659275134&srefix=,aps,279&sr=8-1&language=en_US¤cy=KRW](https://www.amazon.com/-/ko/dp/1492057614/ref=sr_1_1?crid=80HRM2JMHD9&keywords=Learning+SQL&qid=1659275134&srefix=,aps,279&sr=8-1&language=en_US¤cy=KRW)



https://play.google.com/store/books/details/Alan_Beaulieu_Learning_SQL?id=NI3UDwAAQBAJ 영어

https://play.google.com/store/books/details/앨런_볼리외_러닝_SQL?id=gLkIEAAAQBAJ 한국어

EBS 홈페이지 들어가면 있는데 좋다고 함.

입트영

귀트영

<https://5dang.ebs.co.kr/auschoolmain?NaPm=ct%3Dhf2bview|ci%3D0Gy1001FZzHfpCO600-M|tr%3Dsa|et%3Dhf3rbd2w|ba%3D1.0|aa%3D1.0|hkh%3Db03c9555b13a8cdb2e0c58e6759281a4efcd510d#htis>

자료형

1. 문자 데이터

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

varchar 가변길이: 8글자 데이터를 입력할 때 8글자만 저장된다. 주로 사용하는 문자데이터

char(20) /* fixed-length */

varchar(20) /* variable-length*/

2. 텍스트 데이터

아주 긴 문자열을 저장할 때 사용한다.

text 주로 사용하는 텍스트 데이터

tinytext(성적 같은 것들, abcd등급같이 두 글자를 넘지 않는 것들 - mySQL에만 있음.)

3. 숫자 데이터

int 주로 사용하는 숫자 자료형

tinyint

smallint

bigint

4. 날짜 데이터

timestamp - 현재 날짜와 시간을 자동입력

datetime - 날짜와 시간(예: 생년월일)

테이블 작성

1. 설계 (디자인)

테이블에 저장할 적절한 항목들과 그 항목들을 저장할 데이터 형과 크기를 설계

이름, 주소, 전화번호, 성별, 음식...

2. 정제

테이블 작성시에는 기본키 설정이 중요. (PRIMARY KEY)

이름의 경우에는 성과 이름으로 분리

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),
```

```
  country VARCHAR(20),
```

```
  postal_code VARCHAR(20),
```

```
  CONSTRAINT pk_person PRIMARY KEY (person_id)
```

```
);
```

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

<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/4f9de4c0-aa29-4a52-b863-97da05d053dd/sakila-db.zip>

<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/4f9de4c0-aa29-4a52-b863-97da05d053dd/sakila-db.zip>

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

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

꼭 MySQL 마리아DB에서 실행해야됨.

저번 수행작업문을 다시 적용시켜야 됨.

```
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

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

```
ALTER TABLE person MODIFY person_id SMALLINT UNSIGNED AUTO_INCREMENT;
```

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

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

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

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

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

여기서부터 헤이디 타이핑하기 시작

SQL 주석 2가지

```
/* 긴 주석  
   은 이렇게*/
```

```
SELECT person_id, fname, lname, birth_date  
FROM person  
WHERE person_id = 1; -- 한 줄 주석
```

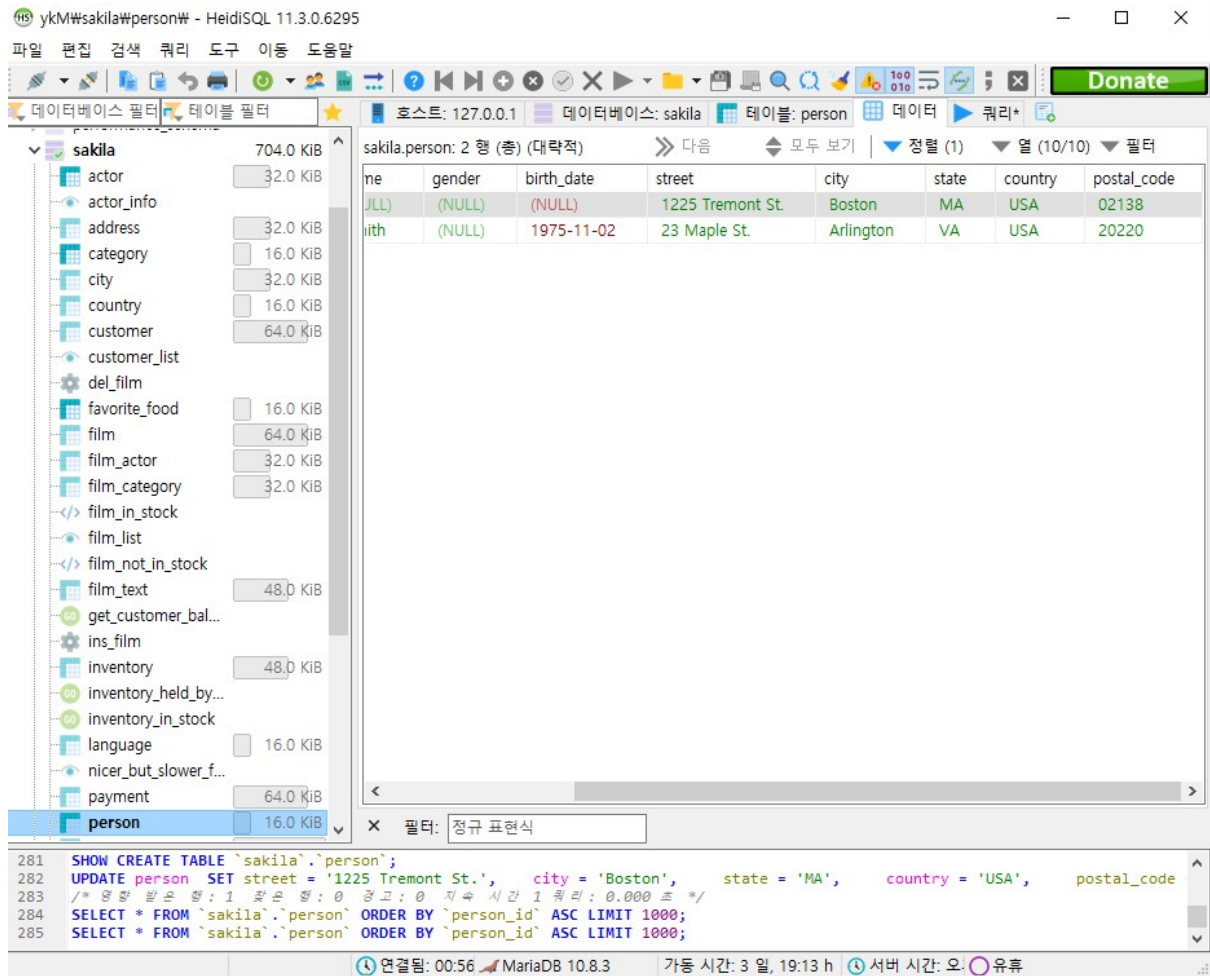
```
INSERT INTO favorite_food (person_id, food)
VALUES (1, 'pizza');
INSERT INTO favorite_food (person_id, food)
VALUES (1, 'cookies');
INSERT INTO favorite_food (person_id, food)
VALUES (1, 'nachos');
```

food 열을 favorite_food이라는 테이블로부터 조건은 person_id = 1인 값만 순서를 food열을 오름차순 정렬하여 조회

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

```
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');
```

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



DELETE FROM person
WHERE person_id = 2;

person 테이블에서 person_id값이 2인 레코드를 삭제.

```

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'
  
```

INSERT INTO person
(person_id, fname, lname, gender, birth_date)
VALUES (1, 'Charles', 'Fulton', 'M', '1968-01-15');

고유 키값이 중복된 데이터를 입력하려고 시도할 때 에러 발생

```
mysql> INSERT INTO favorite_food (person_id, food)
-> VALUES (999, 'lasagna');
ERROR 1452 (23000): Cannot add or update a child row: a foreign key constraint
fails ('bank'. 'favorite_food', CONSTRAINT 'fk_fav_food_person_id' FOREIGN KEY
('person_id') REFERENCES 'person' ('person_id'))
```

존재하지 않는 외래 키 foreign key를 참조할 때 에러 발생

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

열 값 위반, 선언한 데이터형을 벗어났을 때 에러 발생

```
mysql> UPDATE person
-> SET birth_date = 'DEC-21-1980'
-> WHERE person_id = 1;
ERROR 1292 (22007): Incorrect date value: 'DEC-21-1980' for column 'birth_date'
at row 1
```

잘못된 날짜 변환, 날짜의 기본형인 년-월-일로 입력되지 않고 월-일-년으로 입력되어 에러발생

```
mysql> DROP TABLE favorite_food;
Query OK, 0 rows affected (0.56 sec)
mysql> DROP TABLE person;
Query OK, 0 rows affected (0.05 sec)
```

테이블 제거

DROP TABLE 테이블 이름;

DESC - 구조를 자세히 보는 것

DESC customer; // (이젠 이걸 쓰겠다.)

적용해보자. 마리아 DB 들어가서 적용.

<https://s3-us-west-2.amazonaws.com/secure.notion-static.com/2339a2be-1eea-4072-a9bd-22e857da8336/LearningSQLExample.sql>

SOURCE C:/temp/LearningSQLExample.sql;

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

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

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

SELECT - 쿼리 결과에 포함시킬 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 from절을 먼저 이해해야 함.

```
SELECT *
FROM department;
```



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

이 쿼리에서의 FROM은 department라는 하나의 테이블의 모든 열을 결과에 포함하는 것을 나타낸다. * asterisk 문자는 모든 열을 지정한다.

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

또는 하나의 열만 선택하여 결과를 볼 수도 있다.

```
mysql> SELECT 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;
```

emp_id	ACTIVE	emp_id * 3.14159	UPPER(lname)
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

필터: 정규 표현식

숫자나 문자를 그냥 출력

기존열의 값을 계산한 결과를 출력

함수를 사용한 결과

컬럼 별칭

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

AS 생략가능

```
SELECT emp_id 사번,  
       'ACTIVE' 상태,  
       UPPER(lname) 대문자성  
FROM employee;
```

* 진짜 바꾸는 것이 아니라 바뀌서 보여주는 것일 뿐. 원본은 그대로.

중복 제거 DISTINCT

상황에 따라 쿼리가 중복된 행을 반환할 수 있다. 고유한 하나의 값만 남기고 나머지는 제거하여 값을 확인할 수 있다.

```
SELECT cust_id  
FROM account;
```

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

* 진짜 바꾸는 것이 아니라 바뀌서 보여주는 것일 뿐. 원본은 그대로.

FROM 절

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

FROM 절은 쿼리에 사용되는 테이블을 명시할 뿐만 아니라 테이블들을 서로 연결하는 수단도 정의하게 된다.

Permanent Table 영구테이블 create table로 생성된 테이블

Temporary Table 임시테이블 서브 쿼리로 반환된 행들

Virtual Table 가상 테이블 create view로 생성된 테이블

Temporary Table 파상테이블 (← 임시테이블)

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

```
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;
```

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

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

```
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;
```

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';
```

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

```
SELECT d.name, COUNT(e.emp_id)
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 절을 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.

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

```
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

오름차순으로

```
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

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

정렬의 기준이 여러개인 경우는 첫번째 정렬을 마친 값들중 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

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

내림차순으로

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

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

정렬의 기본은 오름차순으로 적시하지 않으면 오름차순 정렬되고

DESC를 적으면 내림차순으로 정렬된다.

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

- 오라클 전용 DB라서 사용 안함.

```
alter session set nls_date_format='RR/MM/DD';
```

```
drop table emp;
```

```
drop table dept;
```

```
CREATE TABLE DEPT
```

```
(DEPTNO number(10),
```

```
DNAME VARCHAR2(14),
```

```
LOC VARCHAR2(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          NUMBER(4) NOT NULL,
```

```
ENAME          VARCHAR2(10),
```

```
JOB            VARCHAR2(9),
```

```
MGR            NUMBER(4) ,
```

```
HIREDATE       DATE,
```

```
SAL            NUMBER(7,2),
```

```
COMM           NUMBER(7,2),
```

```
DEPTNO         NUMBER(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);

commit;

drop table salgrade;

create table salgrade
( grade number(10),
  losal number(10),
  hisal number(10) );

insert into salgrade values(1,700,1200);
insert into salgrade values(2,1201,1400);
insert into salgrade values(3,1401,2000);
insert into salgrade values(4,2001,3000);
insert into salgrade values(5,3001,9999);

commit;

```

<https://www.w3schools.com/>

배우면 수료증도 줌.

Oracle Live SQL

 <https://livesql.oracle.com/apex/f?p=590:1000>

오라클 ID: u8yes@naver.com

비번은 대문자가 들어감.

1) 마리아DB에서 쿼리 입력하면 됨.

```

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');
```

2) 쿼리에 입력

```
CREATE TABLE EMP (
EMPNO      INT(4) NOT NULL,
ENAME      VARCHAR(10),
JOB        VARCHAR(9),
MGR        INT(4) ,
HIREDATE   DATE,
SAL        INT(7),
COMM       INT(7),
DEPTNO     INT(2) );
```

```
INSERT INTO EMP VALUES (7839,'KING','PRESIDENT',NULL,'81-11-17',5000,NULL,10);
INSERT INTO EMP VALUES (7698,'BLAKE','MANAGER',7839,'81-05-01',2850,NULL,30);
INSERT INTO EMP VALUES (7782,'CLARK','MANAGER',7839,'81-05-09',2450,NULL,10);
INSERT INTO EMP VALUES (7566,'JONES','MANAGER',7839,'81-04-01',2975,NULL,20);
INSERT INTO EMP VALUES (7654,'MARTIN','SALESMAN',7698,'81-09-10',1250,1400,30);
INSERT INTO EMP VALUES (7499,'ALLEN','SALESMAN',7698,'81-02-11',1600,300,30);
INSERT INTO EMP VALUES (7844,'TURNER','SALESMAN',7698,'81-08-21',1500,0,30);
INSERT INTO EMP VALUES (7900,'JAMES','CLERK',7698,'81-12-11',950,NULL,30);
INSERT INTO EMP VALUES (7521,'WARD','SALESMAN',7698,'81-02-23',1250,500,30);
INSERT INTO EMP VALUES (7902,'FORD','ANALYST',7566,'81-12-11',3000,NULL,20);
INSERT INTO EMP VALUES (7369,'SMITH','CLERK',7902,'80-12-11',800,NULL,20);
INSERT INTO EMP VALUES (7788,'SCOTT','ANALYST',7566,'82-12-22',3000,NULL,20);
INSERT INTO EMP VALUES (7876,'ADAMS','CLERK',7788,'83-01-15',1100,NULL,20);
INSERT INTO EMP VALUES (7934,'MILLER','CLERK',7782,'82-01-11',1300,NULL,10);
```

QUIZ 시작.

1.

사원 테이블에서 사원 번호와 이름과 월급을 출력해 보겠습니다.

내가 답변 한 것.

호스트: 127.0.0.1 데이터베이스: ba

```

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

X 필터: 정규 표현식

대소문자, 들여쓰기 다 가리지 않고 인식함.

```

/*
SELECT EMPNO, ENAME, SAL
FROM EMP;

SELECT empno, ename, sal
FROM emp;

SELECT empno, ename, sal FROM emp;

SELECT empno, ename, sal
    FROM emp;
*/

SELECT empno, ename, sal
    FROM emp;

```

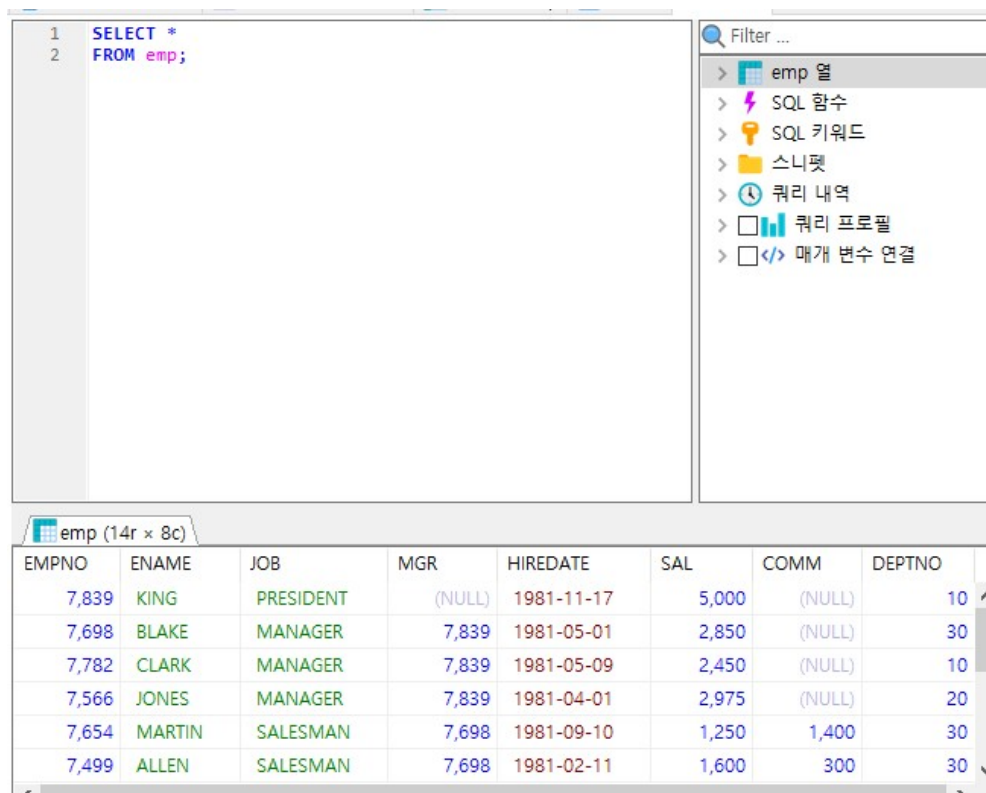
QUIZ 시작.

2.

사원 테이블을 모든 열(column)들을 전부 출력해 보겠습니다.

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

내가 답변 한 것.



The screenshot shows a SQL IDE interface. The top pane contains the SQL query: `SELECT * FROM emp;`. The bottom pane displays the results of the query in a table format. The table has 8 columns: EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, and DEPTNO. The results show 7 rows of employee data.

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

Quiz 003

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

내가 답변 한 것.

호스트: 127.0.0.1 데이터베이스: bank

```

1 SELECT EMPNO AS '사원번호',
2        ENAME AS '이름',
3        SAL AS '사원 Salary'
4 FROM emp;
5

```

emp (14r x 3c)

사원번호	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

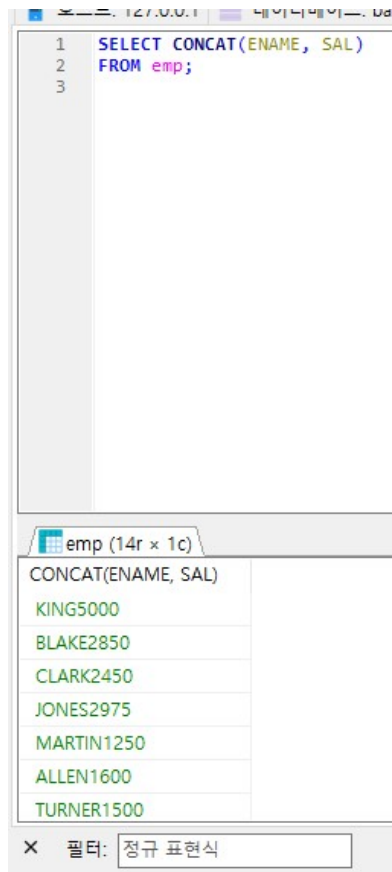
× 필터: 정규 표현식

Quiz 004

사원 테이블의 이름과 월급을 서로 붙여서 출력해 보겠습니다.

KING5000
BLAKE2850
CLARK2450
JONES2975
:

내가 답변 한 것.



- 오라클에서만 쓸 수 있는 방법(붙이기)

(MySQL 에서 || 는 참을 판별하는 용도로만 사용)

```
SELECT ename || sal
FROM emp;
```

Quiz 005

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

내가 답변 한 것.

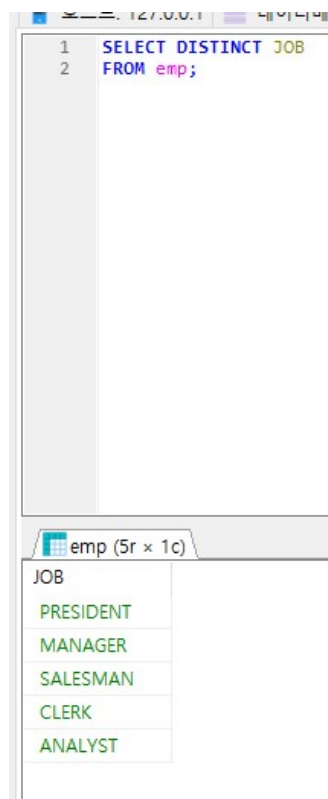
오스노: 127.0.0.1 데이터베이스: bank 테이블: emp 데이터	
1	SELECT CONCAT(ENAME, ' 의 직업은 ', JOB, ' 입니다')
2	FROM emp;
emp (14r x 1c)	
CONCAT(ENAME, ' 의 직업은 ', JOB, ' 입니다')	
KING 의 직업은 PRESIDENT 입니다	
BLAKE 의 직업은 MANAGER 입니다	
CLARK 의 직업은 MANAGER 입니다	
JONES 의 직업은 MANAGER 입니다	
MARTIN 의 직업은 SALESMAN 입니다	
ALLEN 의 직업은 SALESMAN 입니다	
TURNER 의 직업은 SALESMAN 입니다	

Quiz 006

사원 테이블에서 직업을 출력하는데 중복된 데이터를 제외하고 출력해 보겠습니다.

JOB
SALESMAN
CLERK
ANALYST
MANAGER
PRESIDENT

내가 답변 한 것.



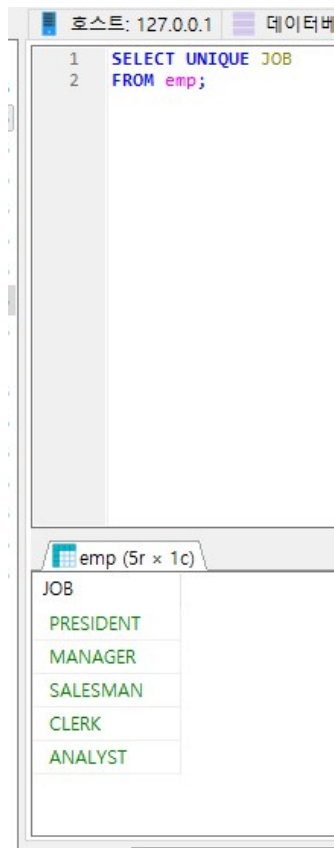
The screenshot shows a SQL IDE window with a query editor and a results pane. The query editor contains the following SQL statement:

```
1 SELECT DISTINCT JOB
2 FROM emp;
```

The results pane shows the output of the query, which is a list of distinct job titles from the emp table. The results are displayed in a table with one column, JOB, and five rows:

JOB
PRESIDENT
MANAGER
SALESMAN
CLERK
ANALYST

새로운 방법



Quiz 007

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

내가 답변 한 것.

1	/*SELECT ENAME, SAL
2	FROM emp
3	ORDER BY SAL;*/
4	
5	SELECT ENAME, SAL
6	FROM emp
7	ORDER BY SAL DESC;

emp (14r x 2c)	
ENAME	SAL
TURNER	1,500
MILLER	1,300
WARD	1,250
MARTIN	1,250
ADAMS	1,100
JAMES	950
SMITH	800

Quiz 008

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

내가 답변 한 것.

1	SELECT ENAME, DEPTNO, SAL
2	FROM emp
3	ORDER BY DEPTNO ASC, SAL DESC;
4	

emp (14r x 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