

코딩역량 강화 과정 Java & SQL

2일차 - 2

장은실

Contents

01 SELECT ... FROM 문

02 SELECT ... FROM ... WHERE 문

03 GROUP BY ... HAVING 문

1-1 SQL 문의 개요

- SQL(Structured Query Language, 구조화된 질의 언어) 문
 - 데이터베이스에서 사용되는 일종의 공통 언어
 - NCITS(국제표준화위원회)에서 ANSI/ISO SQL이라는 명칭의 SQL 표준을 관리하고 있음
 - 1992년에 제정된 ANSI-92 SQL과 1999년에 제정된 ANSI-99 SQL을 대부분의 DBMS 회사에서 SQL 표준으로 사용하고 있음
 - 각 회사는 ANSI-92/99 SQL의 표준을 준수하면서도 자신의 제품 특성을 반영한 SQL에 별도의 이름을 붙임
 - MySQL에서는 그냥 SQL, 오라클에서는 PL/SQL, SQL Server에서는 Transact SQL(T-SQL) 사용

1-2 SQL 문의 종류

- DML(Data Manipulation Language, 데이터 조작어)
 - 데이터를 검색 및 삽입, 수정, 삭제하는데 사용하는 언어
 - SELECT 문, INSERT, UPDATE, DELETE 문 등
- DDL(Data Definition Language, 데이터 정의어)
 - 데이터베이스, 테이블, 뷰, 인덱스 등의 데이터베이스 개체를 생성, 삭제, 변경하는 데 사용하는 언어
 - CREATE, DROP, ALTER, TRUNCATE 문 등
- DCL(Data Control Language, 데이터 제어어)
 - 사용자에게 어떤 권한을 부여하거나 빼앗을 때 사용하는 언어
 - GRANT, REVOKE, DENY 문 등

1-3 USE 문

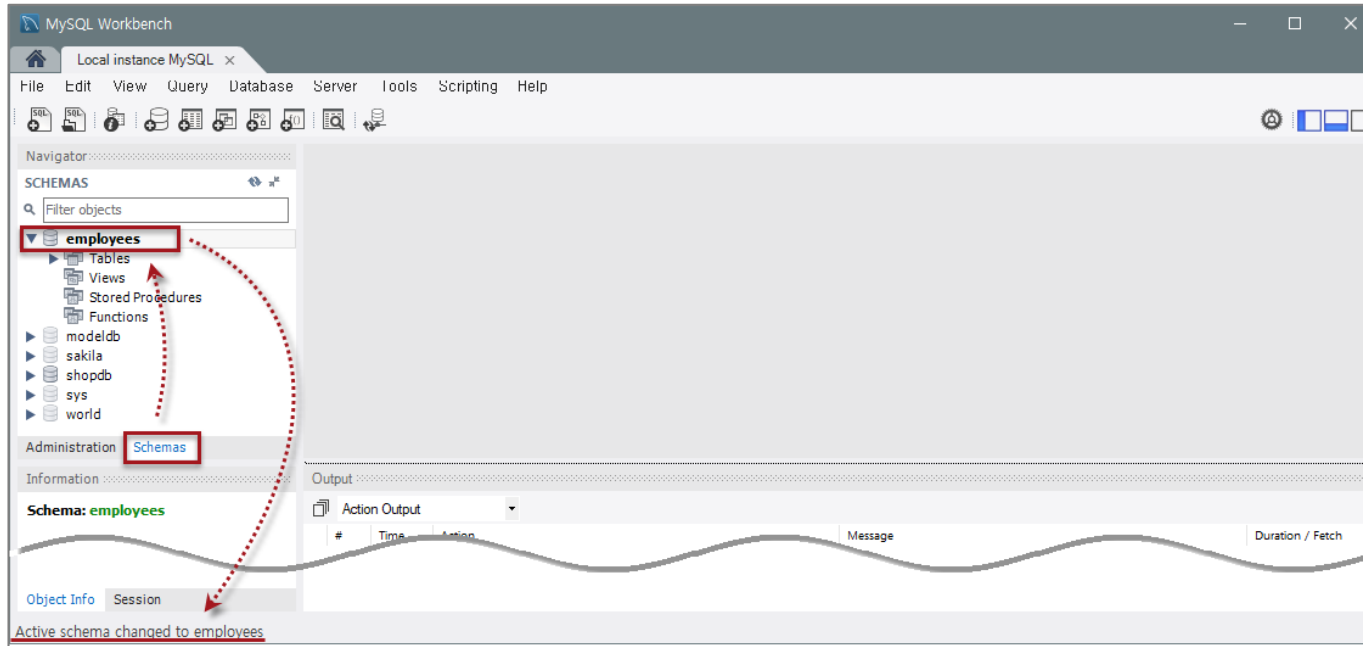
- 현재 사용하는 데이터베이스를 지정하거나 변경하는 구문 형식

USE 데이터베이스이름;

- employees 데이터베이스를 사용하려면 다음과 같이 입력

USE employees;

- Workbench에서 데이터베이스를 지정하는 방법



1-3 USE 문

- 쿼리 창을 연 후 자신이 작업할 데이터베이스가 선택되어 있는지 먼저 확인하는 습관을 들여야 함

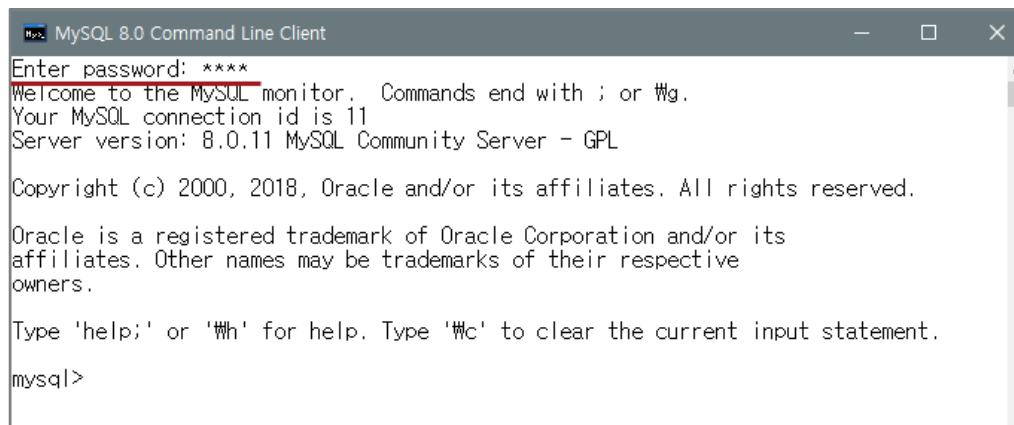
```
USE mysql;  
SELECT * FROM employees;
```

Output				
Action Output				
	#	Time	Action	Message
✓	1	10:35:58	USE mysql	0 row(s) affected
✗	2	10:35:58	SELECT * FROM employees LIMIT 0, 1000	Error Code: 1146. Table 'mysql.employees' doesn't exist

개체의 이름을 정확히 모를 때 데이터 검색하기

1 명령 줄 모드로 MySQL 서버에 접속하기

1-1 명령 줄 모드로 MySQL 서버에 접속



```
MySQL 8.0 Command Line Client
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.11 MySQL Community Server - GPL

Copyright (c) 2000, 2018, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

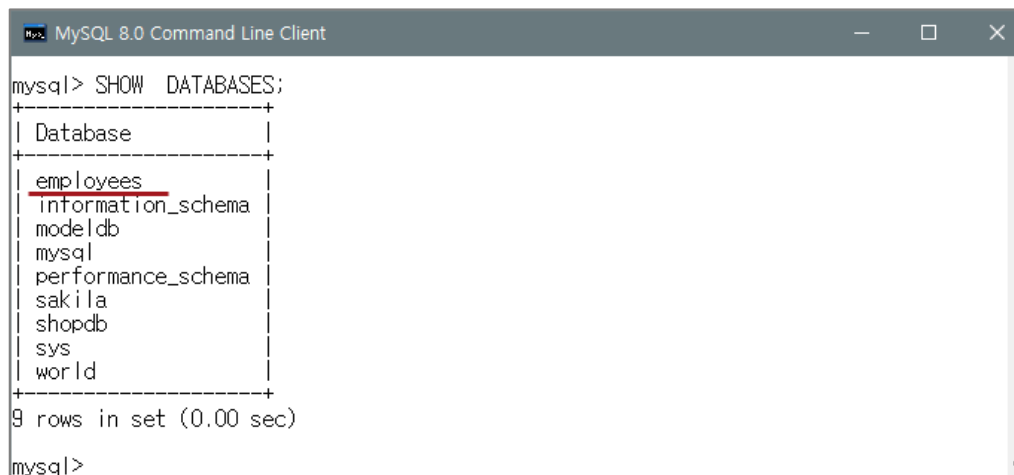
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

2 개체 이름을 조회한 후 원하는 작업 하기

2-1 현재 서버에 어떤 데이터베이스가 있는지 조회

SHOW DATABASES;



```
MySQL 8.0 Command Line Client
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| employees |
| information_schema |
| modeldb |
| mysql |
| performance_schema |
| sakila |
| shopdb |
| sys |
| world |
+-----+
9 rows in set (0.00 sec)

mysql>
```

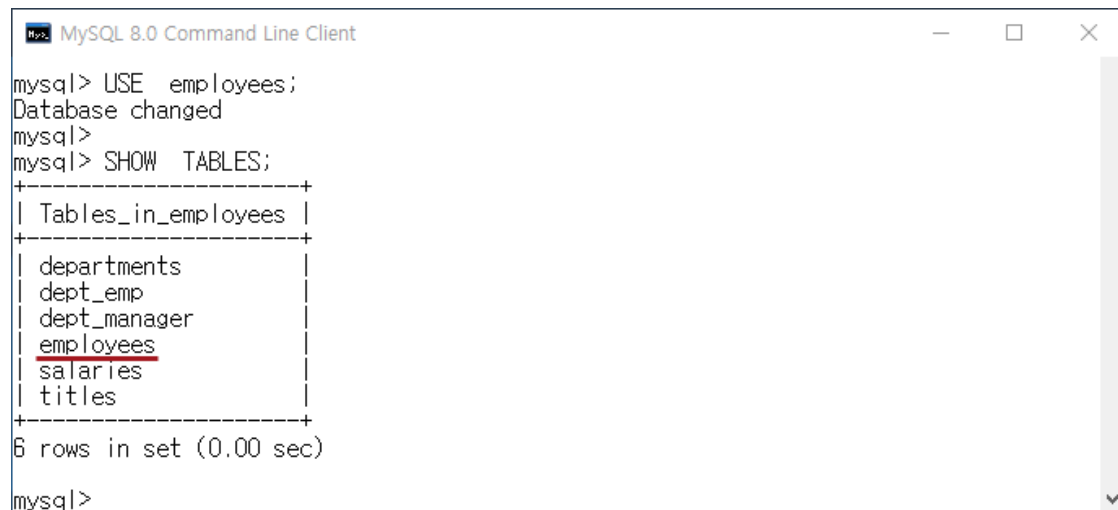
개체의 이름을 정확히 모를 때 데이터 검색하기

2-2 employees를 앞으로 사용할 데이터베이스로 지정

```
USE employees;
```

2-3 현재 서버에 어떤 데이터베이스가 있는지 조회

```
SHOW TABLES;
```



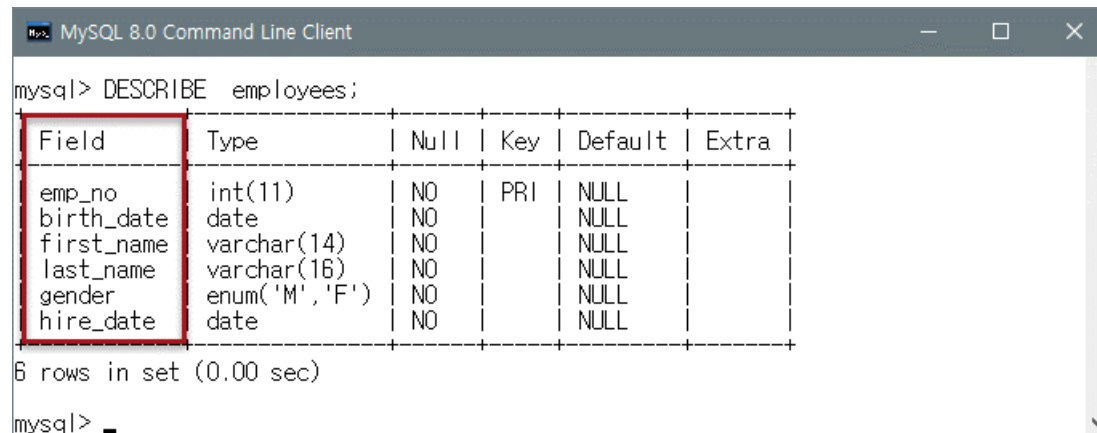
```
MySQL 8.0 Command Line Client
mysql> USE employees;
Database changed
mysql>
mysql> SHOW TABLES;
+-----+
| Tables_in_employees |
+-----+
| departments          |
| dept_emp             |
| dept_manager         |
| employees           |
| salaries             |
| titles               |
+-----+
6 rows in set (0.00 sec)

mysql>
```


개체의 이름을 정확히 모를 때 데이터 검색하기

2-4 employees 테이블의 열에는 무엇이 있는지 확인

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



```
mysql> DESCRIBE employees;
```

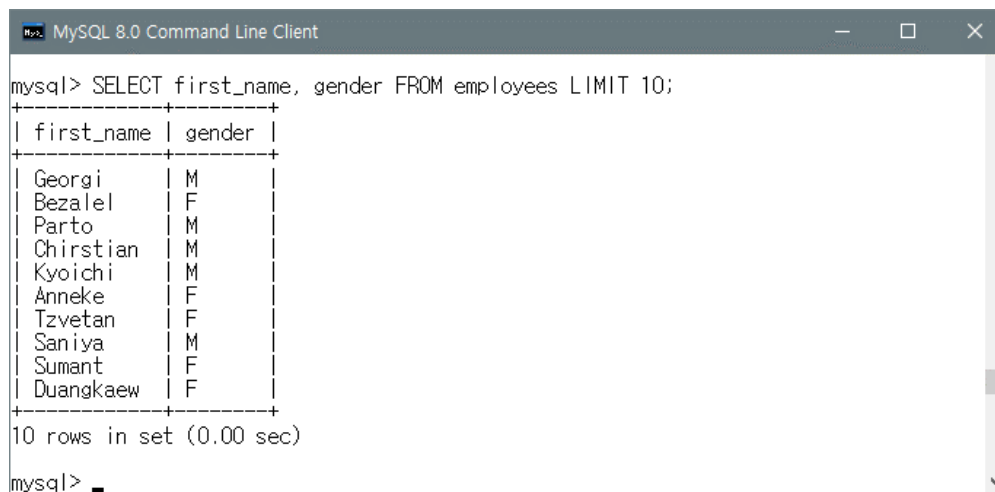
Field	Type	Null	Key	Default	Extra
emp_no	int(11)	NO	PRI	NULL	
birth_date	date	NO		NULL	
first_name	varchar(14)	NO		NULL	
last_name	varchar(16)	NO		NULL	
gender	enum('M', 'F')	NO		NULL	
hire_date	date	NO		NULL	

6 rows in set (0.00 sec)

```
mysql>
```

2-5 최종적으로 원하는 열 조회

```
SELECT first_name, gender FROM employees LIMIT 10;
```



```
mysql> SELECT first_name, gender FROM employees LIMIT 10;
```

first_name	gender
Georgi	M
Bezalel	F
Parto	M
Chirstjan	M
Kyoichi	M
Anneke	F
Tzvetan	F
Saniya	M
Sumant	F
Duangkaew	F

10 rows in set (0.00 sec)

```
mysql>
```

cookDB 샘플 데이터베이스의 개요

■ cookDB 소개



cookDB 샘플 데이터베이스

1 cookDB 생성하기

1-1 cookDB를 생성하는 쿼리문 입력

```
DROP DATABASE IF EXISTS cookDB; -- 만약 cookDB가 존재하면 우선 삭제한다.  
CREATE DATABASE cookDB;
```

1-2 회원 테이블과 구매 테이블을 생성하는 쿼리문 입력

```
USE cookDB;  
CREATE TABLE userTBL -- 회원 테이블  
( userID CHAR(8) NOT NULL PRIMARY KEY, -- 사용자 아이디(PK)  
  userName VARCHAR(10) NOT NULL, -- 이름  
  birthYear INT NOT NULL, -- 출생 연도  
  addr CHAR(2) NOT NULL, -- 지역(경기, 서울, 경남 식으로 2글자만 입력)  
  mobile1 CHAR(3), -- 휴대폰의 국번(011, 016, 017, 018, 019, 010 등)  
  mobile2 CHAR(8), -- 휴대폰의 나머지 번호(하이픈 제외)  
  height SMALLINT, -- 키  
  mDate DATE -- 회원 가입일  
);  
CREATE TABLE buyTBL -- 구매 테이블  
( num INT AUTO_INCREMENT NOT NULL PRIMARY KEY, -- 순번(PK)  
  userID CHAR(8) NOT NULL, -- 아이디(FK)  
  prodName CHAR(6) NOT NULL, -- 물품  
  groupName CHAR(4), -- 분류  
  price INT NOT NULL, -- 단가  
  amount SMALLINT NOT NULL, -- 수량  
  FOREIGN KEY (userID) REFERENCES userTBL (userID)  
);
```

1-3 회원 테이블과 구매 테이블에 데이터 삽입

```
INSERT INTO userTBL VALUES ('YJS', '유재석', 1972, '서울', '010', '11111111', 178, '2008-8-8');
INSERT INTO userTBL VALUES ('KHD', '강호동', 1970, '경북', '011', '22222222', 182, '2007-7-7');
INSERT INTO userTBL VALUES ('KKJ', '김국진', 1965, '서울', '019', '33333333', 171, '2009-9-9');
INSERT INTO userTBL VALUES ('KYM', '김용만', 1967, '서울', '010', '44444444', 177, '2015-5-5');
INSERT INTO userTBL VALUES ('KJD', '김제동', 1974, '경남', NULL, NULL, 173, '2013-3-3');
INSERT INTO userTBL VALUES ('NHS', '남희석', 1971, '충남', '016', '66666666', 180, '2017-4-4');
INSERT INTO userTBL VALUES ('SDY', '신동엽', 1971, '경기', NULL, NULL, 176, '2008-10-10');
INSERT INTO userTBL VALUES ('LHJ', '이휘재', 1972, '경기', '011', '88888888', 180, '2006-4-4');
INSERT INTO userTBL VALUES ('LKK', '이경규', 1960, '경남', '018', '99999999', 170, '2004-12-12');
INSERT INTO userTBL VALUES ('PSH', '박수홍', 1970, '서울', '010', '00000000', 183, '2012-5-5');
```

```
INSERT INTO buyTBL VALUES (NULL, 'KHD', '운동화', NULL, 30, 2);
INSERT INTO buyTBL VALUES (NULL, 'KHD', '노트북', '전자', 1000, 1);
INSERT INTO buyTBL VALUES (NULL, 'KYM', '모니터', '전자', 200, 1);
INSERT INTO buyTBL VALUES (NULL, 'PSH', '모니터', '전자', 200, 5);
INSERT INTO buyTBL VALUES (NULL, 'KHD', '청바지', '의류', 50, 3);
INSERT INTO buyTBL VALUES (NULL, 'PSH', '메모리', '전자', 80, 10);
INSERT INTO buyTBL VALUES (NULL, 'KJD', '책', '서적', 15, 5);
INSERT INTO buyTBL VALUES (NULL, 'LHJ', '책', '서적', 15, 2);
INSERT INTO buyTBL VALUES (NULL, 'LHJ', '청바지', '의류', 50, 1);
INSERT INTO buyTBL VALUES (NULL, 'PSH', '운동화', NULL, 30, 2);
INSERT INTO buyTBL VALUES (NULL, 'LHJ', '책', '서적', 15, 1);
INSERT INTO buyTBL VALUES (NULL, 'PSH', '운동화', NULL, 30, 2);
```

1-4 두 테이블에 삽입된 데이터 확인

```
SELECT * FROM userTBL;
SELECT * FROM buyTBL;
```

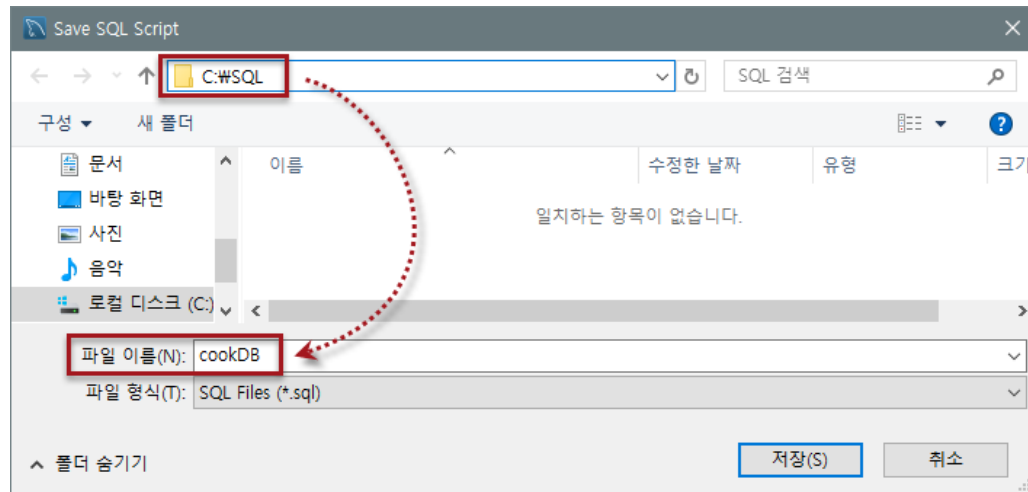
	userID	userName	birthYear	addr	mobile1	mobile2	height	mDate
▶	KHD	강호동	1970	경북	011	22222222	182	2007-07-07
	KJD	김제동	1974	경남	NULL	NULL	173	2013-03-03
	KKJ	김국진	1965	서울	019	33333333	171	2009-09-09
	KYM	김용만	1967	서울	010	44444444	177	2015-05-05
	LHJ	이휘재	1972	경기	011	88888888	180	2006-04-04
	LKK	이경규	1960	경남	018	99999999	170	2004-12-12
	NHS	남희석	1971	충남	016	66666666	180	2017-04-04
	PSH	박수홍	1970	서울	010	00000000	183	2012-05-05
	SDY	신동엽	1971	경기	NULL	NULL	176	2008-10-10
	YJS	유재석	1972	서울	010	11111111	178	2008-08-08
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

	num	userID	prodName	groupName	price	amount
▶	1	KHD	운동화	NULL	30	2
	2	KHD	노트북	전자	1000	1
	3	KYM	모니터	전자	200	1
	4	PSH	모니터	전자	200	5
	5	KHD	청바지	의류	50	3
	6	PSH	메모리	전자	80	10
	7	KJD	책	서적	15	5
	8	LHJ	책	서적	15	2
	9	LHJ	청바지	의류	50	1
	10	PSH	운동화	NULL	30	2
	11	LHJ	책	서적	15	1
	12	PSH	운동화	NULL	30	2
*	NULL	NULL	NULL	NULL	NULL	NULL

cookDB 샘플 데이터베이스

2 cookDB 저장하기

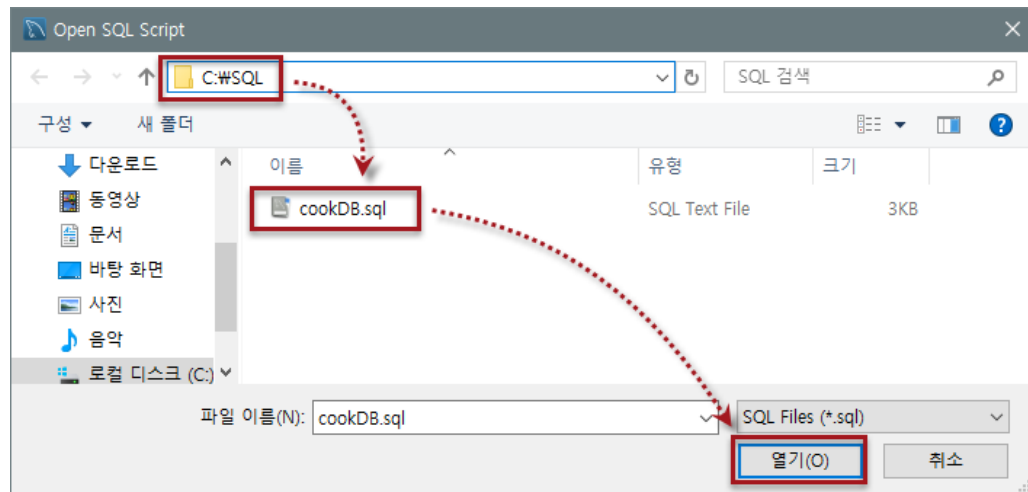
2-1 'cookDB.sql' 저장



3 cookDB 초기화하기

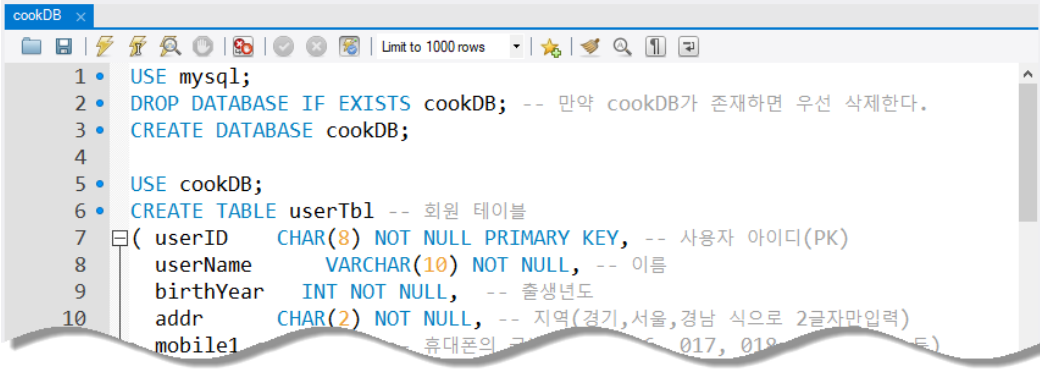
3-1 열려 있는 쿼리 창 모두 닫기

3-2 C:\#SQL\cookDB.sql 파일 <열기>



cookDB 샘플 데이터베이스

3-3 SQL 문을 실행해 cookDB 초기화



```
1 • USE mysql;
2 • DROP DATABASE IF EXISTS cookDB; -- 만약 cookDB가 존재하면 우선 삭제한다.
3 • CREATE DATABASE cookDB;
4
5 • USE cookDB;
6 • CREATE TABLE userTbl -- 회원 테이블
7 • ( userID CHAR(8) NOT NULL PRIMARY KEY, -- 사용자 아이디(PK)
8 •   userName VARCHAR(10) NOT NULL, -- 이름
9 •   birthYear INT NOT NULL, -- 출생년도
10 •  addr CHAR(2) NOT NULL, -- 지역(경기, 서울, 경남 식으로 2글자만입력)
    mobile1 CHAR(11) NOT NULL -- 휴대폰의 국번(010, 017, 018 등)
```

3-4 왼쪽 내비게이터에 cookDB가 보이지 않으면 [Refresh All] 선택

1-4 SELECT 문의 형식

- MySQL의 도움말에 나오는 SELECT 문의 형식

```
SELECT      *      |  { [DISTINCT] {{column_name | expr }                {[AS] [ alias ]}, ...} }  
FROM        table_name  
WHERE       search_condition      [ {AND | OR } , search_condition ...] ;
```

[구문 설명]

- 키워드 SELECT 다음에는 조회하려고 하는 컬럼 이름(또는 표현 식)을 기술
 - 모든 컬럼을 조회하는 경우 *asterisk 기호를 사용할 수 있음 → 컬럼은 생성된 순서대로 표시됨
 - 여러 컬럼을 조회하는 경우 각 컬럼 이름은 쉼표로 구분(마지막 컬럼 이름 다음에는 쉼표를 사용하지 않음)
 - 조회 결과는 기술된 컬럼 이름 순서로 표시
- 키워드 FROM 다음에는 조회 대상 컬럼이 포함된 테이블 이름을 기술
- 키워드 WHERE 다음에는 행을 선택하는 조건을 기술
 - 제한 조건을 여러 개 포함할 수 있으며, 각각의 제한 조건은 논리 연산자로 연결
 - 제한 조건을 만족시키는 행(논리 연산 결과가 TRUE 인 행)들만 Result set에 포함

1-4 SELECT 문의 형식

■ 요약된 SELECT 문의 형식

```
SELECT select_expr  
  [FROM table_references]  
  [WHERE where_condition]  
  [GROUP BY {col_name | expr | position}]  
  [HAVING where_condition]  
  [ORDER BY {col_name | expr | position}]
```

■ 더 요약된 SELECT 문의 형식

```
SELECT 열이름  
FROM 테이블이름  
WHERE 조건
```

■ SELECT 문의 실행 순서

FROM → WHERE → GROUP BY → HAVING → SELECT → DISTINCT → ORDER BY

ON → JOIN

CUBE | ROLLUP

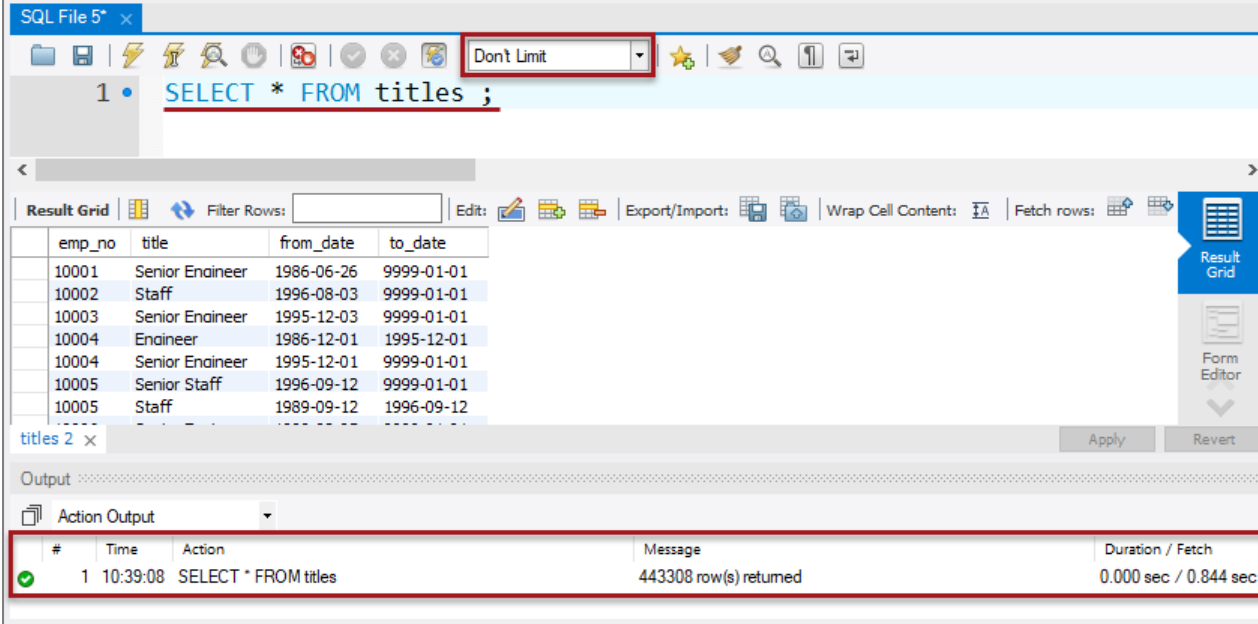
TOP

1. FROM : 조회 테이블 확인
2. WHERE : 데이터 추출 조건 확인
3. GROUP BY : 컬럼 그룹화
4. HAVING : 그룹화 조건
5. SELECT : 데이터 추출
6. ORDER BY : 데이터 순서 정렬

1-5 SELECT ... FROM 문

■ 모든 열 검색

```
SELECT * FROM titles;
```



The screenshot shows a SQL IDE interface. The top toolbar includes a dropdown menu set to "Don't Limit". The SQL File editor contains the query: `SELECT * FROM titles ;`. Below the editor, the Result Grid displays the following data:

emp_no	title	from_date	to_date
10001	Senior Engineer	1986-06-26	9999-01-01
10002	Staff	1996-08-03	9999-01-01
10003	Senior Engineer	1995-12-03	9999-01-01
10004	Engineer	1986-12-01	1995-12-01
10004	Senior Engineer	1995-12-01	9999-01-01
10005	Senior Staff	1996-09-12	9999-01-01
10005	Staff	1989-09-12	1996-09-12

Below the Result Grid, the Action Output pane shows the execution details for the query:

#	Time	Action	Message	Duration / Fetch
1	10:39:08	SELECT * FROM titles	443308 row(s) returned	0.000 sec / 0.844 sec

- 초록색 아이콘: 쿼리가 정상적으로 실행된 상태를 나타냄
- 1: 실행한 쿼리의 순번을 나타냄
- Action: 실행한 쿼리문이 표시됨
- Message : SELECT 문으로 조회한 행의 개수가 표시
- Duration/Fetch: Duration은 SQL 문이 실행되는 데 걸린 시간(초), Fetch는 데이터를 테이블에서 가져오는 데 걸린 시간(초)을 나타냄

1-5 SELECT ... FROM 문

- 여러 개의 열을 가져오고 싶으면 쉼표(,)로 구분

```
SELECT first_name, last_name, gender FROM employees;
```

The screenshot displays a database client interface. At the top, there's a toolbar with options like 'Filter Rows', 'Export', 'Wrap Cell Content', and 'Fetch rows'. Below this is a 'Result Grid' showing the results of the SQL query. The grid has columns for 'first_name', 'last_name', and 'gender'. The data rows are as follows:

first_name	last_name	gender
Georgi	Facello	M
Bezael	Simmel	F
Parto	Bamford	M
Chirstian	Koblick	M
Kvoichi	Maliniak	M
Anneke	Preusis	F
Tzvetan	Zielinski	F

Below the result grid, there's a tab labeled 'employees 6' and a 'Read Only' indicator. At the bottom, there's an 'Output' section with a dropdown menu set to 'Action Output'. The output log shows the following entry:

#	Time	Action	Message	Duration / Fetch
✓ 1	10:46:44	SELECT first_name, last_name, gender FROM employees LI...	1000 row(s) returned	0.000 sec / 0.000 sec

1-5 SELECT ... FROM 문

- 현재 선택된 데이터베이스가 employees라면 다음 두 쿼리는 동일

```
SELECT * FROM employees.titles;  
SELECT * FROM titles;
```

- 원하는 열만 검색

```
SELECT first_name FROM employees;
```

The screenshot shows a database client interface. At the top, there's a toolbar with icons for 'Result Grid', 'Filter Rows', 'Export', 'Wrap Cell Content', and 'Fetch rows'. Below the toolbar, a table displays the results of a query. The table has one column, 'first_name', and several rows of names. To the right of the table, there's a vertical sidebar with buttons for 'Result Grid' and 'Form Editor'. Below the table, there's a tab labeled 'employees 5' with a 'Read Only' indicator. At the bottom, there's an 'Output' section with a dropdown menu set to 'Action Output'. Below this, a table shows the execution log.

#	Time	Action	Message	Duration / Fetch
✓ 1	10:42:08	SELECT first_name FROM employees LIMIT 0, 1000	1000 row(s) returned	0.000 sec / 0.000 sec

2-1 WHERE 절

- SELECT ... FROM 문에 WHERE 절을 추가하면 특정한 조건을 만족하는 데이터만 조회할 수 있음

SELECT 열이름 FROM 테이블이름 WHERE 조건식;

- WHERE 절 없이 cookDB의 회원 테이블(userTBL) 조회

```
USE cookDB;  
SELECT * FROM userTBL;
```

- 원 테이블(userTBL)에서 강호동의 정보만 조회

```
SELECT * FROM userTBL WHERE userName = '강호동';
```

[illegible]

2-1 WHERE 절 조건

- WHERE 절에 조건으로 사용할 수 있는 술어

술어	연산자	예
비교	=, <>, <, <=, >, >=	price < 20000
범위	BETWEEN	price BETWEEN 10000 AND 20000
집합	IN, NOT IN	price IN (10000, 20000, 30000)
패턴	LIKE	bookname LIKE '축구의 역사'
NULL	IS NULL, IS NOT NULL	price IS NULL
복합조건	AND, OR, NOT	(price < 20000) AND (bookname LIKE '축구의 역사')

2-2 조건 연산자와 관계 연산자

- 회원 테이블에서 1970년 이후에 출생했고 키가 182cm 이상인 사람의 아이디와 이름을 조회

```
SELECT userID, userName FROM userTBL WHERE birthYear >= 1970 AND height >= 182;
```

- 1970년 이후에 출생했거나 키가 182cm 이상인 사람의 아이디와 이름 조회

```
SELECT userID, userName FROM userTBL WHERE birthYear >= 1970 OR height >= 182;
```

2-3 BETWEEN ... AND, IN(), LIKE 연산자

- 회원 테이블에서 키가 180~182cm인 사람 조회

```
SELECT userName, height FROM userTBL WHERE height >= 180 AND height <= 182;
```

- 위 쿼리문은 BETWEEN ... AND 연산자를 사용하여 다음과 같이 작성

```
SELECT userName, height FROM userTBL WHERE height BETWEEN 180 AND 182;
```

- 지역이 경남 또는 충남 또는 경북인 사람은 OR 연산자를 사용하여 조회

```
SELECT userName, addr FROM userTBL WHERE addr='경남' OR addr='충남' OR addr='경북';
```

- 이산적인(discrete) 값을 조회할 때는 IN() 연산자 사용

```
SELECT userName, addr FROM userTBL WHERE addr IN ('경남', '충남', '경북');
```

- 성이 김 씨인 회원의 이름과 키 조회

```
SELECT userName, height FROM userTBL WHERE userName LIKE '김%';
```

- 맨 앞의 한 글자가 무엇이든 상관없고 그다음이 '경규'인 사람 조회

```
SELECT userName, height FROM userTBL WHERE userName LIKE '_경규';
```


2-4 ORDER BY 절

- 가입한 순서대로 회원 출력(기본적으로 오름차순(ascending)으로 정렬)

```
SELECT userName, mDate FROM userTBL ORDER BY mDate;
```

	userName	mDate
▶	이경규	2004-12-12
	이회재	2006-04-04
	강호동	2007-07-07
	유재석	2008-08-08
	신동엽	2008-10-10
	김국진	2009-09-09
	박수홍	2012-05-05
	김제동	2013-03-03
	김용만	2015-05-05
	남희석	2017-04-04

- 내림차순(descending)으로 정렬(열 이름 뒤에 DESC를 넣음)

```
SELECT userName, mDate FROM userTBL ORDER BY mDate DESC;
```

- 정렬 기준을 2개로 설정하고 정렬

```
SELECT userName, height FROM userTBL ORDER BY height DESC, userName ASC;
```

2-5 DISTINCT 키워드

- 회원 테이블에서 회원들의 거주 지역이 몇 곳인지 출력

```
SELECT addr FROM userTBL;
```

	addr
▶	경북
	경남
	서울
	서울
	경기
	경남
	충남
	서울
	경기
	서울

- 회원 테이블에서 회원들의 거주 지역이 몇 곳인지 출력(ORDER BY 절 사용)

```
SELECT addr FROM userTBL ORDER BY addr;
```

	addr
▶	경기
	경기
	경남
	경남
	경북
	서울
	서울
	서울
	서울
	충남

2-5 DISTINCT 키워드

- 중복 지역을 하나만 출력

```
SELECT DISTINCT addr FROM userTBL;
```

	addr
▶	경북
	경남
	서울
	경기
	충남

2-6 LIMIT 절

- 입사일이 오래된 직원 5명의 emp_no(사원번호) 조회(' Don't Limit' 선택)

```
USE employees;  
SELECT emp_no, hire_date FROM employees  
ORDER BY hire_date ASC;
```

The screenshot shows a SQL IDE window with a dropdown menu set to 'Don't Limit'. The SQL query is: `USE employees; SELECT emp_no, hire_date FROM employees ORDER BY hire_date ASC;`. The result grid displays a list of employee IDs and hire dates. The 'Output' pane shows the execution log with the message '300024 row(s) returned' highlighted in red.

emp_no	hire_date
110022	1985-01-01
110085	1985-01-01
110183	1985-01-01
110303	1985-01-01
110511	1985-01-01
296777	1985-01-01
297266	1985-02-01
234495	1985-02-01
235733	1985-02-01

#	Time	Action	Message	Duration / Fetch
1	17:53:16	USE employees	0 row(s) affected	0.015 sec
2	17:53:16	SELECT emp_no, hire_date FROM employees ORD...	300024 row(s) returned	0.360 sec / 0.172 sec

- 상위의 N개만 출력하는 LIMIT 절 사용

```
SELECT emp_no, hire_date FROM employees  
ORDER BY hire_date ASC  
LIMIT 5;
```

The screenshot shows the same SQL IDE window with the 'LIMIT 5' query. The result grid displays only the first 5 rows of the sorted data. The 'Output' pane shows the execution log with the message '5 row(s) returned' highlighted in red.

emp_no	hire_date
110022	1985-01-01
110511	1985-01-01
110303	1985-01-01
110085	1985-01-01
110183	1985-01-01
NULL	NULL

#	Time	Action	Message	Duration / Fetch
1	17:55:53	SELECT emp_no, hire_date FROM employees ORD...	5 row(s) returned	0.375 sec / 0.000 sec

2-6 LIMIT 절

- ' LIMIT 시작, 개수' 형식으로 조회

```
SELECT emp_no, hire_date FROM employees  
ORDER BY hire_date ASC  
LIMIT 0, 5; -- 0번부터 5개
```

3-1 GROUP BY 절

- SELECT 문의 형식 중에서 GROUP BY ... HAVING 절의 위치

```
SELECT select_expr  
  [FROM table_references]  
  [WHERE where_condition]  
  [GROUP BY {col_name | expr | position}]  
  [HAVING where_condition]  
  [ORDER BY {col_name | expr | position}]
```

- cookDB의 구매 테이블 (buyTBL)에서 아이디(userID)마다 구매한 물건의 개수(amount)를 조회하는 쿼리문

```
USE cookDB;  
SELECT userID, amount FROM buyTBL ORDER BY userID;
```

The screenshot displays a database management interface. At the top, there's a 'Result Grid' section showing a table with columns 'userID' and 'amount'. The data rows are:

userID	amount
KHD	2
KHD	1
KHD	3
PSH	10
PSH	2
PSH	2

Below the result grid, there's a tab labeled 'buyTbl 19'. Underneath, an 'Output' section shows the 'Action Output' for the executed query. The log entry for the query 'SELECT userID, amount FROM buyTbl ORDER BY userID' shows '12 row(s) returned', which is highlighted with a red box.

#	Time	Action	Message	Duration / Fetch
1	19:59:55	USE cookDB	0 row(s) affected	0.000 sec
2	19:59:55	SELECT userID, amount FROM buyTbl ORDER BY userID	12 row(s) returned	0.000 sec / 0.000 sec

3-1 GROUP BY 절

- 같은 아이디(userID)끼리 GROUP BY 절로 묶은 후 SUM() 함수로 구매 개수(amount)를 합치는 방식

```
SELECT userID, SUM(amount) FROM buyTBL GROUP BY userID;
```

	userID	SUM(amount)
▶	KHD	6
	KJD	5
	KYM	1
	LHJ	4
	PSH	19

- 별칭을 사용하여 열 이름을 이해하기 좋게 변경

```
SELECT userID AS '사용자 아이디', SUM(amount) AS '총 구매 개수'  
FROM buyTBL GROUP BY userID;
```

	사용자 아이디	총 구매 개수
▶	KHD	6
	KJD	5
	KYM	1
	LHJ	4
	PSH	19

- 구매액의 총합

```
SELECT userID AS '사용자 아이디', SUM(price * amount) AS '총구매액'  
FROM buyTBL GROUP BY userID;
```

	사용자 아이디	총 구매액
▶	KHD	1210
	KJD	75
	KYM	200
	LHJ	95
	PSH	1920

3-2 집계 함수

- 자주 사용되는 집계 함수

표 5-1 집계 함수의 종류

함수	설명
AVG()	평균을 구한다.
MIN()	최솟값을 구한다.
MAX()	최댓값을 구한다.
COUNT()	행의 개수를 센다.
COUNT(DISTINCT)	행의 개수를 센다(중복은 1개만 인정).
STDEV()	표준편차를 구한다.
VAR_SAMP()	분산을 구한다.

- 전체적으로 한 번 구매할 때마다 평균 몇 개를 구매했는지 조회

```
USE cookDB;  
SELECT AVG(amount) AS '평균 구매 개수' FROM buyTBL;
```

	평균 구매 개수
▶	2.9167

3-2 집계 함수

- 회원별로 한 번 구매할 때마다 평균적으로 몇 개를 구매했는지 조회(GROUP BY 절 사용)

```
SELECT userID, AVG(amount) AS '평균 구매 개수'  
FROM buyTBL GROUP BY userID;
```

	userID	평균 구매 개수
▶	KHD	2.0000
	KJD	5.0000
	KYM	1.0000
	LHJ	1.3333
	PSH	4.7500

- 가장 키가 큰 회원과 가장 키가 작은 회원의 이름과 키 출력

```
SELECT userName, MAX(height), MIN(height) FROM userTBL;
```

	userName	MAX(height)	MIN(height)
	강호동	183	170

- GROUP BY 절을 활용하여 수정

```
SELECT userName, MAX(height), MIN(height)  
FROM userTBL GROUP BY userName;
```

	userName	MAX(height)	MIN(height)
▶	강호동	182	182
	김제동	173	173
	김국진	171	171
	김용만	177	177
	이휘재	180	180
	이경규	170	170
	남희석	180	180
	박수홍	183	183
	신동엽	176	176
	유재석	178	178

3-2 집계 함수

- 서브쿼리와 조합하여 다시 실행

```
SELECT userName, height  
FROM userTBL  
WHERE height = (SELECT MAX(height) FROM userTBL)  
OR height = (SELECT MIN(height) FROM userTBL);
```

	userName	height
▶	이경규	170
	박수홍	183

- 휴대폰이 있는 회원의 수(의도와 다르게 전체 회원이 조회됨)

```
SELECT COUNT( * ) FROM userTBL;
```

- 휴대폰이 있는 회원만 세려면 휴대폰 열 이름(mobile1)을 지정해야 함

```
SELECT COUNT(mobile1) AS '휴대폰이 있는 사용자' FROM userTBL;
```

	휴대폰이 있는 사용자
▶	8

3-3 HAVING 절

- WHERE 절에서는 집계함수를 사용 할 수 없다.
- HAVING 절은 집계함수를 가지고 조건비교를 할 때 사용하며, HAVING절은 GROUP BY절과 함께 사용이 된다.

■ 아이디별 총구매액 구하기

```
USE cookDB;  
SELECT userID AS '사용자', SUM(price * amount) AS '총구매액'  
FROM buyTBL  
GROUP BY userID;
```

	사용자	총구매액
▶	KHD	1210
	KJD	75
	KYM	200
	LHJ	95
	PSH	1920

■ 총 구매액이 1000 이상인 회원에게만 사은품을 증정하고 싶다면?

```
SELECT userID AS '사용자', SUM(price * amount) AS '총구매액'  
FROM buyTBL  
WHERE SUM(price * amount) > 1000  
GROUP BY userID;
```

Output					
Action Output					
#	Time	Action	Message	Duration / Fetch	
❌ 1	09:51:14	SELECT userID AS '사용자', SUM(price*amount) AS '총구...	Error Code: 1111. Invalid use of group function	0.000 sec	

3-3 HAVING 절

- HAVING 절을 사용하여 다시 작성

```
SELECT userID AS '사용자', SUM(price * amount) AS '총구매액'  
FROM buyTBL  
GROUP BY userID  
HAVING SUM(price * amount) > 1000;
```

	사용자	총구매액
▶	KHD	1210
	PSH	1920

- 총 구매액이 적은 회원 순으로 정렬(ORDER BY 절 사용)

```
SELECT userID AS '사용자', SUM(price * amount) AS '총구매액'  
FROM buyTBL  
GROUP BY userID  
HAVING SUM(price * amount) > 1000  
ORDER BY SUM(price * amount);
```

SQL 문

employees 데이터베이스 employees 테이블

1. 모든 레코드 조회하기

사원(employees) 테이블에서 모든 사원 정보를 emp_no 순으로 조회하는 SQL 문을 작성해 주세요.

```
SELECT * FROM employees ORDER BY emp_no;
```

2. 사원 아이디와 이름 조회하기

사원(employees) 테이블에서 emp_no, first_name, last_name 정보를 emp_no 순으로 조회하는 SQL 문을 작성해 주세요.

```
SELECT emp_no, first_name, last_name FROM employees ORDER BY emp_no;
```

3. 역순 정렬하기

사원(employees) 테이블에서 emp_no, first_name, last_name 정보를 emp_no 역순으로 조회하는 SQL 문을 작성해 주세요.

```
SELECT emp_no, first_name, last_name FROM employees ORDER BY emp_no DESC;
```

SQL 문

employees 데이터베이스 employees 테이블

4. 여러 기준으로 정렬하기

사원(employees) 테이블에서 emp_no, first_name, hire_date 정보를 조회하는 SQL 문을 작성해 주세요. 단, first_name 순으로 조회하되 first_name이 같은 경우 나중에 고용(hire_date)된 순으로 출력합니다.

```
SELECT emp_no, first_name, hire_date FROM employees ORDER BY first_name ASC, hire_date DESC;
```

5. 상위 n개 레코드 조회하기

사원(employees) 테이블에서 가장 먼저 고용된 first_name, last_name 정보를 조회하는 SQL 문을 작성해 주세요. 단, 고용된 날짜가 같은 경우 first_name 순으로 1명만 출력합니다.

```
SELECT first_name, last_name FROM employees ORDER BY hire_date ASC, first_name ASC LIMIT 1;
```

6. 남성 사원 조회하기

사원(employees) 테이블에서 남성 사원의 emp_no, last_name 정보를 emp_no 순으로 조회하는 SQL 문을 작성해 주세요.

```
SELECT emp_no, last_name FROM employees WHERE gender="M" ORDER BY emp_no;
```

SQL 문

7. "Nahid"인 여성 사원 조회

사원(employees) 테이블에서 first_name이 "Nahid"인 여성 사원의 last_name 정보를 이름순으로 조회하는 SQL 문을 작성해 주세요.

```
SELECT last_name FROM employees WHERE gender="F" And first_name="Nahid" ORDER BY last_name;
```

8. 그룹명이 없는 유저 아이디

cookdb 데이터베이스 buyTBL 테이블

buyTBL 테이블에서 그룹명(Group Name)이 없는 아이디(userID) 정보를 아이디 순으로 조회하는 SQL 문을 작성해 주세요.

```
SELECT userID FROM buyTBL WHERE GroupName IS NULL ORDER BY userID;
```

9. 그룹명이 있는 유저 아이디

buyTBL 테이블에서 그룹명(Group Name)이 있는 아이디(userID) 정보를 아이디 순으로 조회하는 SQL 문을 작성해 주세요.

```
SELECT userID FROM buyTBL WHERE GroupName IS NOT NULL ORDER BY userID;
```

SQL 문

cookdb 데이터베이스 buyTBL 테이블

10. NULL 처리하기

buyTBL 테이블에서 그룹명(GroupName)이 없는 아이디(userID) 정보를 아이디 순으로 조회하는 SQL 문을 작성해 주세요. 단, NULL이라는 기호 대신 "No name"으로 표시해 주세요.

```
SELECT userID, prodName, IFNULL(GroupName,"No name") AS GroupName  
FROM buyTBL  
WHERE GroupName IS NULL  
ORDER BY userID
```

11. 최대값 구하기

buyTBL 테이블에서 가격(price)이 가장 비싼 제품명(prodName)을 조회하는 SQL 문을 작성해 주세요.

```
SELECT prodName, MAX(price) AS price FROM buyTBL;
```

12. 최소값 구하기

buyTBL 테이블에서 가격(price)이 가장 싼 제품명(prodName)을 조회하는 SQL 문을 작성해 주세요.

```
SELECT prodName, MIN(price) AS price FROM buyTBL;
```


SQL 문

cookdb 데이터베이스 buyTBL 테이블

13. 판매 건수 구하기

buyTBL 테이블에서 노트북 제품의 판매 건수를 조회하는 SQL 문을 작성해 주세요.

```
SELECT COUNT(*) AS COUNT FROM buyTBL WHERE prodName="노트북";
```

14. 중복 제거하기

buyTBL 테이블에서 판매된 제품(prodName)이 몇가지 종류인지 조회하는 SQL 문을 작성해 주세요. 단, 제품명이 NULL인 경우는 집계하지 않으며 중복되는 이름은 하나로 칩니다.

```
SELECT DISTINCT prodName FROM buyTBL WHERE prodName IS NOT NULL;
```

15. 제품별 판매 건수

buyTBL 테이블에서 제품별 판매 건수를 제품명 순으로 조회하는 SQL 문을 작성해 주세요.

```
SELECT prodName, COUNT(*) AS COUNT  
FROM buyTBL  
GROUP BY prodName  
ORDER BY prodName
```

SQL 문

cookdb 데이터베이스 buyTBL 테이블

16. 판매 건수에 따른 제품명 조회

buyTBL 테이블에서 3건 이상의 판매 건수가 있는 제품명 순으로 조회하는 SQL 문을 작성해 주세요.

```
SELECT prodName, COUNT(*) AS COUNT  
FROM buyTBL  
WHERE prodName IS NOT NULL  
GROUP BY prodName  
HAVING COUNT >= 3  
ORDER BY prodName
```

데이터 삽입, 수정, 삭제



1-2 INSERT 문

- INSERT 문

- 테이블에 데이터를 삽입하는 명령어

```
INSERT [INTO] 테이블이름[(열1, 열2, ...)] VALUES (값1, 값2, ...)
```

- INSERT 문에서 테이블 이름 다음에 나오는 열 생략 가능(단 열의 순서 및 개수는 동일해야 함)

```
USE cookDB;  
CREATE TABLE testTBL1 (id int, userName char(3), age int);  
INSERT INTO testTBL1 VALUES (1, '뽀로로', 16);
```

- id와 이름만 입력하고 나이는 입력하고 싶지 않다면

```
INSERT INTO testTBL1 (id, userName) VALUES (2, '크롱');
```

- 열의 순서를 바꾸어 입력하고 싶을 때

```
INSERT INTO testTBL1 (userName, age, id) VALUES ('루피', 14, 3);
```

1-3 UPDATE 문

- UPDATE 문

- 테이블에 입력되어 있는 값을 수정하는 명령어

```
UPDATE 테이블이름  
SET 열1=값1, 열2=값2, ...  
WHERE 조건;
```

- first_name이 'Kyoichi'의 last_name을 '없음'으로 수정

```
USE employees;  
UPDATE employees  
SET last_name = '없음'  
WHERE first_name = 'Kyoichi';
```

- 전체 테이블의 내용을 수정하고 싶을 때는 WHERE 절 생략

```
UPDATE buyTBL  
SET price = price * 1.5;
```

1-4 DELETE 문

- DELETE 문

- 테이블에 데이터를 행 단위로 삭제하는 명령어

```
DELETE FROM 테이블이름 WHERE 조건;
```

- DELETE 문에서 WHERE 절을 생략하면 테이블에 저장된 전체 데이터가 삭제

```
USE cookDB;  
DELETE FROM testTBL4 WHERE Fname = 'Aamer';
```

- Aamer 중에서 상위 몇 건만 삭제하고자 할 때는 추가로 LIMIT 절 사용

```
DELETE FROM testTBL4 WHERE Fname = 'Aamer' LIMIT 5;
```

오늘도 수고하셨습니다.