

05

CHAPTER

데이터 검색과 그룹핑



Contents

01 SELECT ... FROM 문

02 SELECT ... FROM ... WHERE 문

03 GROUP BY ... HAVING 문

학습목표

- 요구 분석과 시스템 설계의 중요성을 이해한다.
- 데이터베이스 모델링의 개념을 이해하고 실제 모델링을 통해 연습한다.
- MySQL Workbench에서 제공하는 데이터베이스 모델링 툴을 실습한다.

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 SELECT 문의 형식

- MySQL의 도움말에 나오는 SELECT 문의 형식

```
SELECT
  [ALL | DISTINCT | DISTINCTROW ]
  [HIGH_PRIORITY]
  [MAX_STATEMENT_TIME = N]
  [STRAIGHT_JOIN]
  [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
  [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
select_expr [, select_expr ...]
[FROM table_references
  [PARTITION partition_list]
[WHERE where_condition]
[GROUP BY {col_name | expr | position}
  [ASC | DESC], ... [WITH ROLLUP]]
[HAVING where_condition]
[ORDER BY {col_name | expr | position}
  [ASC | DESC], ...]
[LIMIT {[offset,] row_count | row_count OFFSET offset}]
[PROCEDURE procedure_name(argument_list)]
[INTO OUTFILE 'file_name'
  [CHARACTER SET charset_name]
  export_options
| INTO DUMPFILE 'file_name'
| INTO var_name [, var_name]]
[FOR UPDATE | LOCK IN SHARE MODE]]
```

1-2 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
```

1-3 USE 문

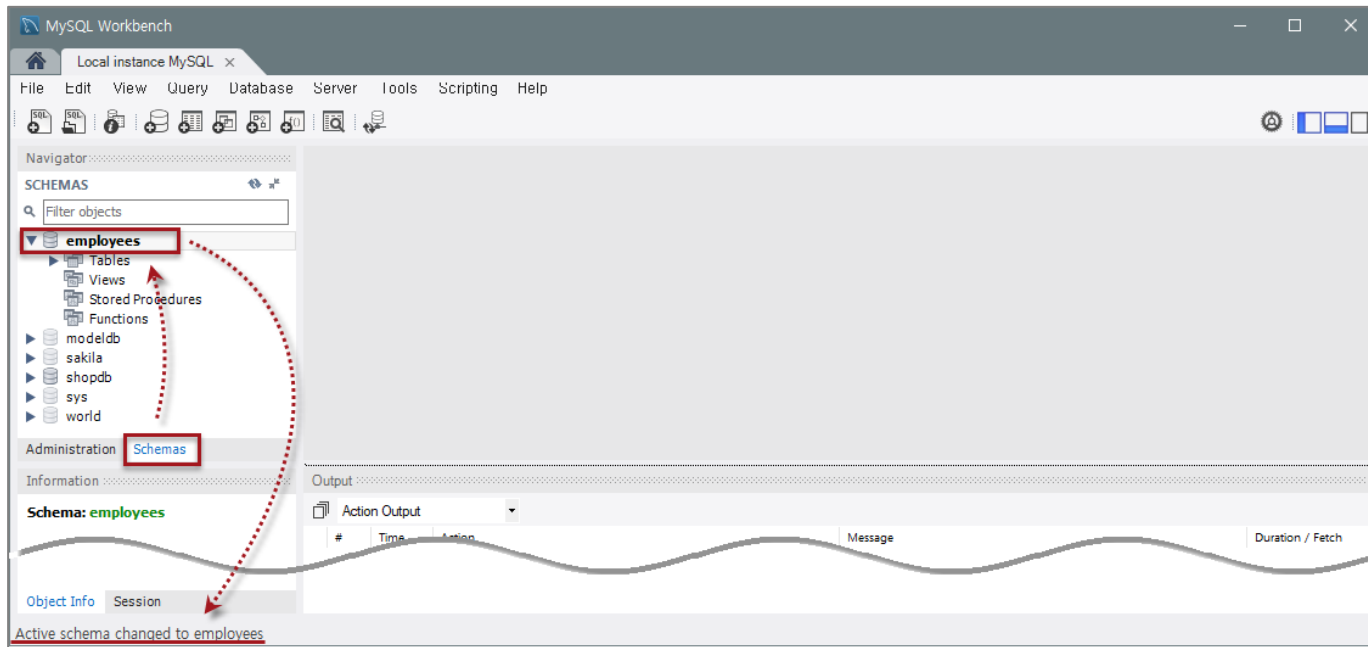
- 현재 사용하는 데이터베이스를 지정하거나 변경하는 구문 형식

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

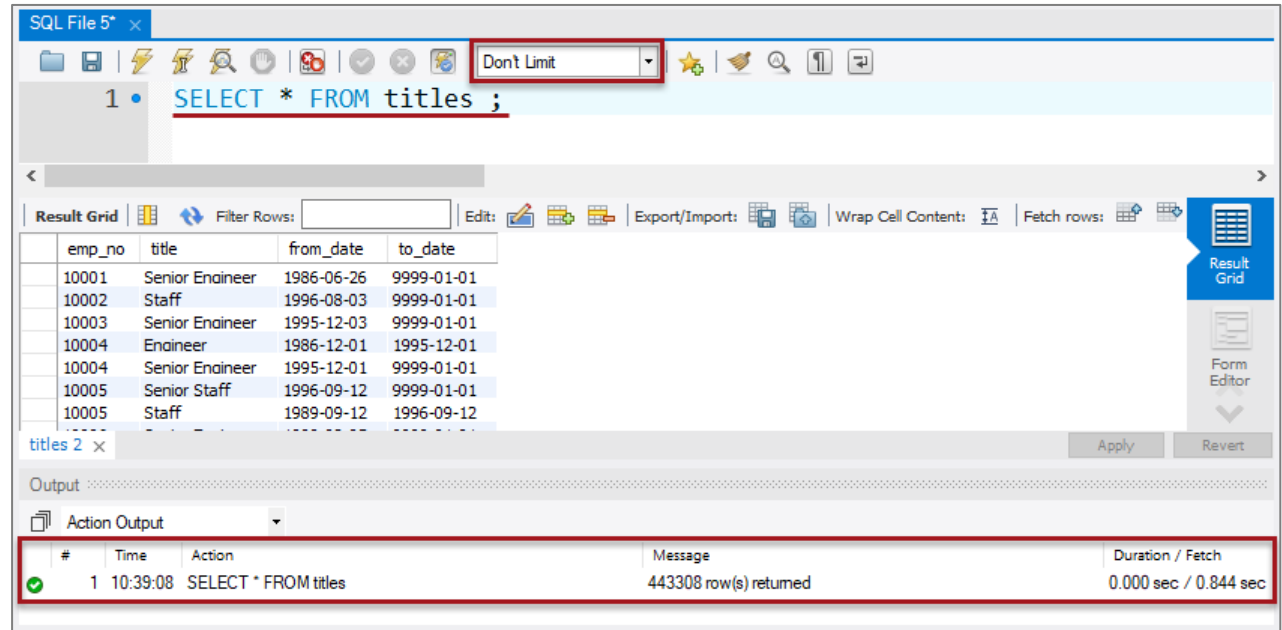
```
USE mysql;  
SELECT * FROM employees;
```

Output				
Action Output				
#	Time	Action	Message	Duration / Fetch
✓ 1	10:35:58	USE mysql	0 row(s) affected	0.000 sec
✗ 2	10:35:58	SELECT * FROM employees LIMIT 0, 1000	Error Code: 1146. Table 'mysql.employees' doesn't exist	0.000 sec

1-4 SELECT ... FROM 문

■ 모든 열 검색

```
SELECT * FROM titles;
```



The screenshot displays the SQL Enterprise Manager interface. The top window, titled 'SQL File 5*', shows the query `SELECT * FROM titles ;` with a red box around the 'Don't Limit' dropdown. Below the query editor is the 'Result Grid' showing a table with columns: emp_no, title, from_date, to_date. The table contains 10 rows of data. At the bottom, the 'Action Output' window shows the execution details for the query, with a red box around the output table.

#	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

1-4 SELECT ... FROM 문

- 여러 개의 열을 가져오고 싶으면 쉼표(,)로 구분

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

The screenshot shows a database query tool interface. At the top, there's a toolbar with options like 'Result Grid', 'Filter Rows', 'Export', 'Wrap Cell Content', and 'Fetch rows'. Below the toolbar is a table with the following data:

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

Below the table, there's a tab labeled 'employees 6' and a 'Read Only' status. At the bottom, there's an 'Output' section with a dropdown menu set to 'Action Output'. It shows a single action with a green checkmark, a timestamp of 10:46:44, the SQL query 'SELECT first_name, last_name, gender FROM employees LI...', a message '1000 row(s) returned', and a duration of '0.000 sec / 0.000 sec'.

1-4 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 query tool interface. At the top, there's a toolbar with options like 'Filter Rows', 'Export', 'Wrap Cell Content', and 'Fetch rows'. Below the toolbar, a table displays the results of a query. The table has a single column 'first_name' and lists several names: Georai, Bezalel, Parto, Chirstian, Kvoichi, Anneke, and Tzvetan. To the right of the table, there's a 'Result Grid' button and a 'Form Editor' button. Below the table, there's a tab labeled 'employees 5' and 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 details of the query.

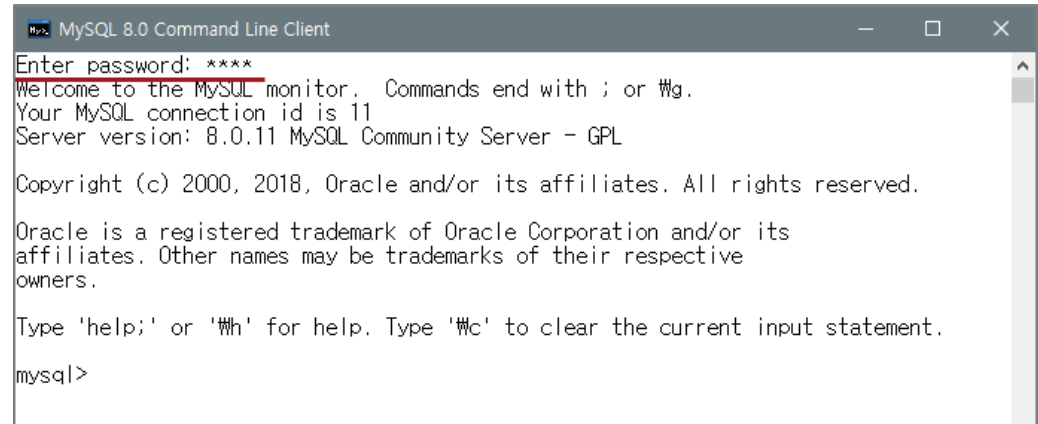
#	Time	Action	Message	Duration / Fetch
✓ 1	10:42:08	SELECT first_name FROM employees LIMIT 0, 1000	1000 row(s) returned	0.000 sec / 0.000 sec

[실습 5-1] 개체의 이름을 정확히 모를 때 데이터 검색하기

교재 158~160p 참고

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>
```

2-2 employees를 앞으로 사용할 데이터베이스로 지정

```
USE employees;
```

2-3 현재 서버에 어떤 데이터베이스가 있는지 조회

```
SHOW TABLES;
```

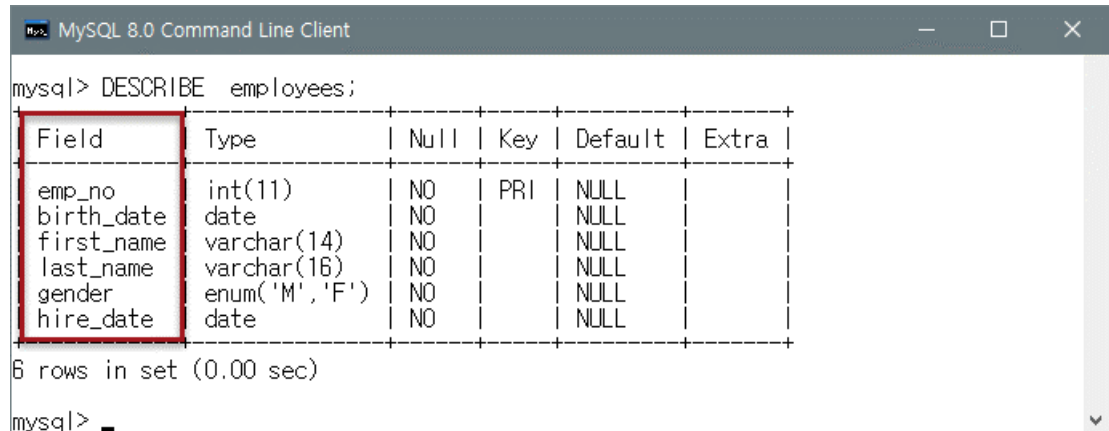


```
MySQL 8.0 Command Line Client
mysql> USE employees;
Database changed
mysql>
mysql> SHOW TABLES;
+-----+
| Tables_in_employees |
+-----+
| departments          |
| dept_emp             |
| dept_manager         |
| employees           |
| salaries             |
| titles               |
+-----+
6 rows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

mysql>
```

2-4 employees 테이블의 열에는 무엇이 있는지 확인

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



```
mysql> DESCRIBE employees;
```

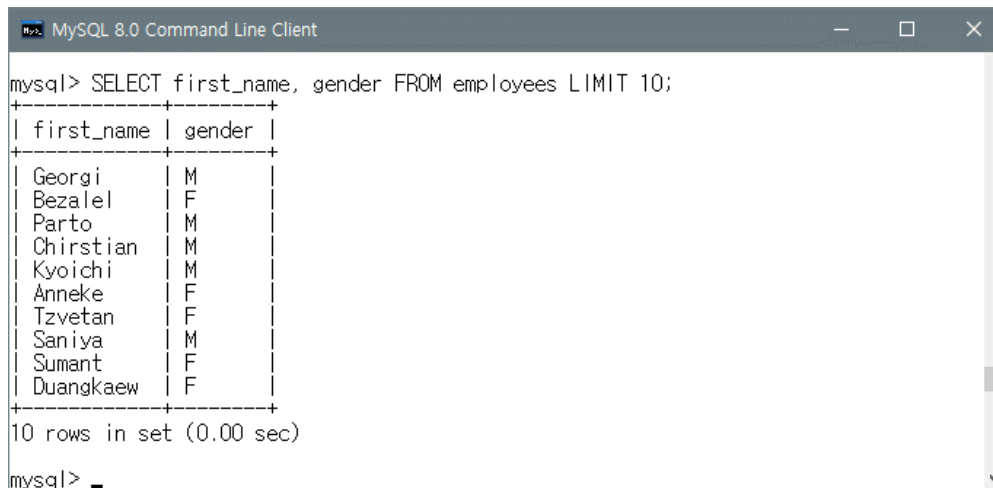
Field	Type	Null	Key	Default	Extra
emp_no	int(11)	NO	PRI	NULL	
birth_date	date	NO		NULL	
first_name	varchar(14)	NO		NULL	
last_name	varchar(16)	NO		NULL	
gender	enum('M','F')	NO		NULL	
hire_date	date	NO		NULL	

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
Chirstian	M
Kyoichi	M
Anneke	F
Tzvetan	F
Saniya	M
Sumant	F
Duangkaew	F

10 rows in set (0.00 sec)

```
mysql>
```

2-1 cookDB 샘플 데이터베이스의 개요

■ cookDB 소개



그림 5-12 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);
```

[실습 5-2] 개체의 이름을 정확히 모를 때 데이터 검색하기

교재 163~167p 참고

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	이회재	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

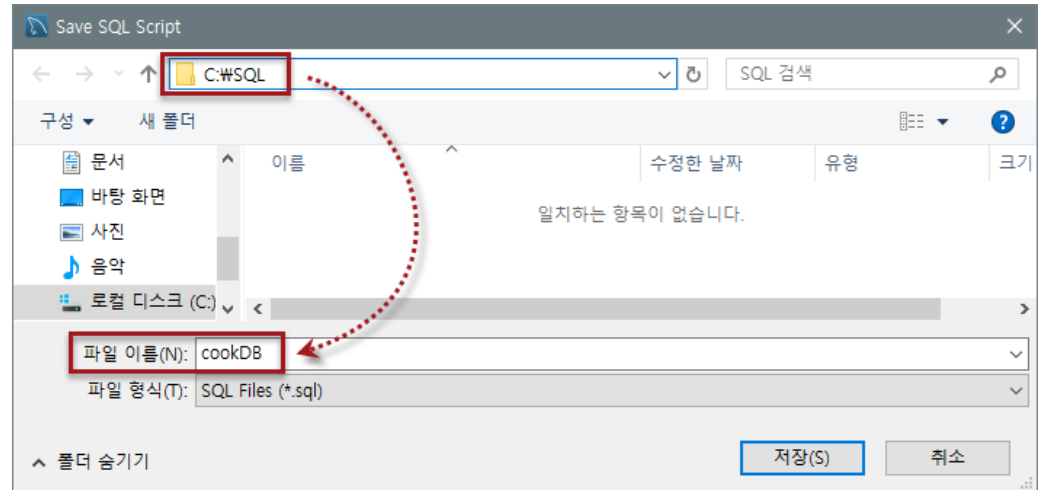
	num	userID	prodName	groupName	price	amount
▶	1	KHD	운동화	NULL	30	2
	2	KHD	노트북	전자	1000	1
	3	KYM	모니터	전자	200	1
	4	PSH	모니터	전자	200	5
	5	KHD	청바지	의류	50	3
	6	PSH	메모리	전자	80	10
	7	KJD	책	서적	15	5
	8	LHJ	책	서적	15	2
	9	LHJ	청바지	의류	50	1
	10	PSH	운동화	NULL	30	2
	11	LHJ	책	서적	15	1
	12	PSH	운동화	NULL	30	2
*	NULL	NULL	NULL	NULL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

[실습 5-2] 개체의 이름을 정확히 모를 때 데이터 검색하기

교재 163~167p 참고

2 cookDB 저장하기

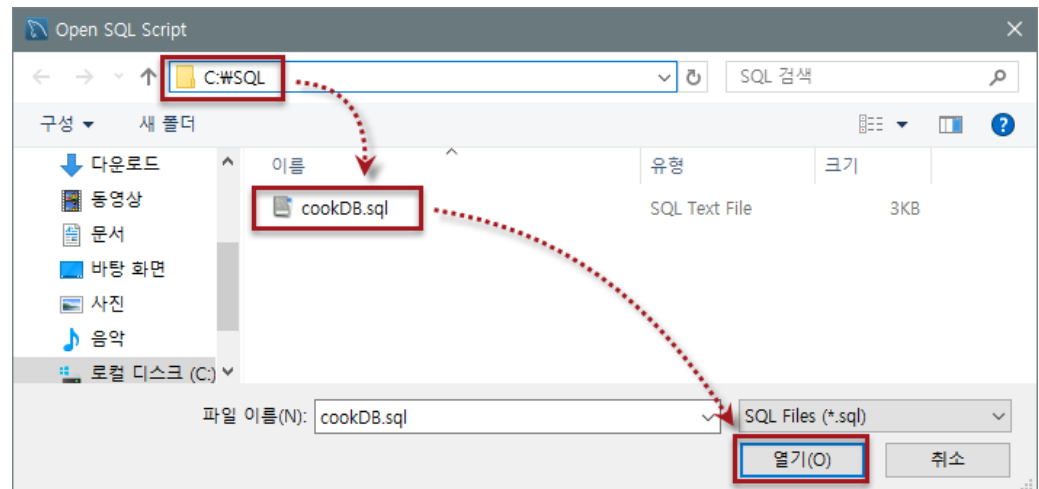
2-1 'cookDB.sql' 저장



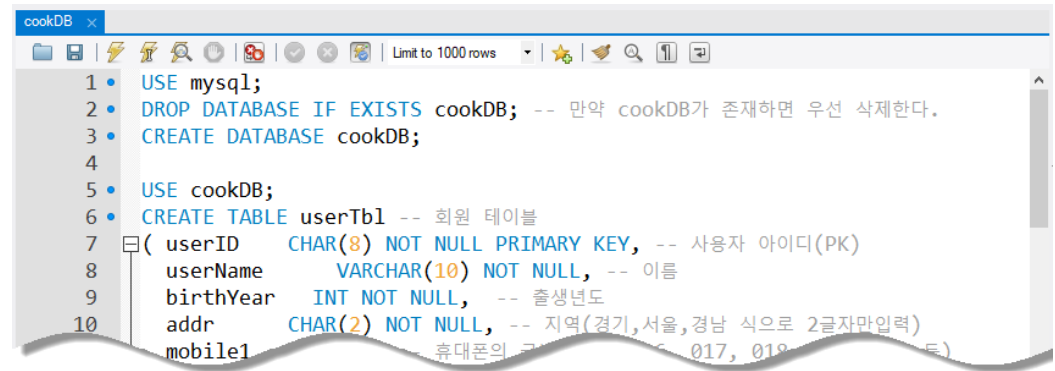
3 cookDB 초기화하기

3-1 열려 있는 쿼리 창 모두 닫기

3-2 C:\SQL\cookDB.sql 파일 <열기>



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(10) NOT NULL -- 휴대폰의 국번(010, 011, 017, 018 등)
```

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

2-2 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';
```

	userID	userName	birthYear	addr	mobile1	mobile2	height	mDate
▶	KHD	강호동	1970	경북	011	22222222	182	2007-07-07
✱	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2-3 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;
```

2-4 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';
```

2-5 서브쿼리와 ANY, ALL, SOME 연산자

- 김용만보다 키가 크거나 같은 사람의 이름과 키 출력

```
SELECT userName, height FROM userTBL WHERE height > 177;
```

```
SELECT userName, height FROM userTBL  
WHERE height > (SELECT height FROM userTBL WHERE userName = '김용만');
```

- 지역이 경기인 사람보다 키가 크거나 같은 사람 추출

```
SELECT userName, height FROM userTBL  
WHERE height >= (SELECT height FROM userTBL WHERE addr = '경기');
```

Output					
Action Output					
#	Time	Action	Message	Duration / Fetch	
1	17:06:31	SELECT userName, height FROM userTBL WHERE height >= (SE...	Error Code: 1242. Subquery returns more than 1 row	0.000 sec	

2-5 서브쿼리와 ANY, ALL, SOME 연산자

- ANY 구문으로 다음과 같이 고친 후 다시 실행

```
SELECT userName, height FROM userTBL  
WHERE height >= ANY (SELECT height FROM userTBL WHERE addr = '경기');
```

	userName	height
▶	강호동	182
	김용만	177
	이휘재	180
	남희석	180
	박수홍	183
	신동엽	176
	유재석	178

- 지역이 경기인 사람보다 키가 크거나 같은 사람 추출

```
SELECT userName, height FROM userTBL  
WHERE height >= ALL (SELECT height FROM userTBL WHERE addr = '경기');
```

	userName	height
▶	강호동	182
	이휘재	180
	남희석	180
	박수홍	183

- >= ANY 대신 = ANY 사용

```
SELECT userName, height FROM userTBL  
WHERE height = ANY (SELECT height FROM userTBL WHERE addr = '경기');
```

	userName	height
▶	이휘재	180
	남희석	180
	신동엽	176

2-5 서브쿼리와 ANY, ALL, SOME 연산자

- '= ANY (서브쿼리)'는 ' IN (서브쿼리)'와 동일

```
SELECT userName, height FROM userTBL  
WHERE height IN (SELECT height FROM userTBL WHERE addr = '경기');
```

2-6 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;
```

2-7 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

2-7 DISTINCT 키워드

- 중복 지역을 하나만 출력

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

	addr
▶	경북
	경남
	서울
	경기
	충남

2-8 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 the following elements:

- SQL File 14*** tab at the top.
- A dropdown menu set to **Don't Limit**.
- SQL code:

```
1 USE employees;  
2 SELECT emp_no, hire_date FROM employees  
3 ORDER BY hire_date ASC;  
4
```
- Result Grid** showing a list of employee records with columns **emp_no** and **hire_date**.
- Output** pane showing the **Action Output** table:

#	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 following elements:

- Result Grid** showing the first 5 rows of the employee list.
- Output** pane showing the **Action Output** table:

#	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

2-8 LIMIT 절

- ' LIMIT 시작, 개수' 형식으로 조회

```
SELECT emp_no, hire_date FROM employees  
ORDER BY hire_date ASC  
LIMIT 0, 5; -- LIMIT 5 OFFSET 0과 동일
```

2-9 CREATE TABLE ... SELECT 문

- CREATE TABLE ... SELECT 구문 형식

CREATE TABLE 새로운테이블 (SELECT 복사할 열 FROM 기존테이블)

- buyTBL 테이블을 buyTBL2 테이블로 복사하는 구문

```
USE cookDB;  
CREATE TABLE buyTBL2 (SELECT * FROM buyTBL);  
SELECT * FROM buyTBL2;
```

- 지정한 일부 열만 복사

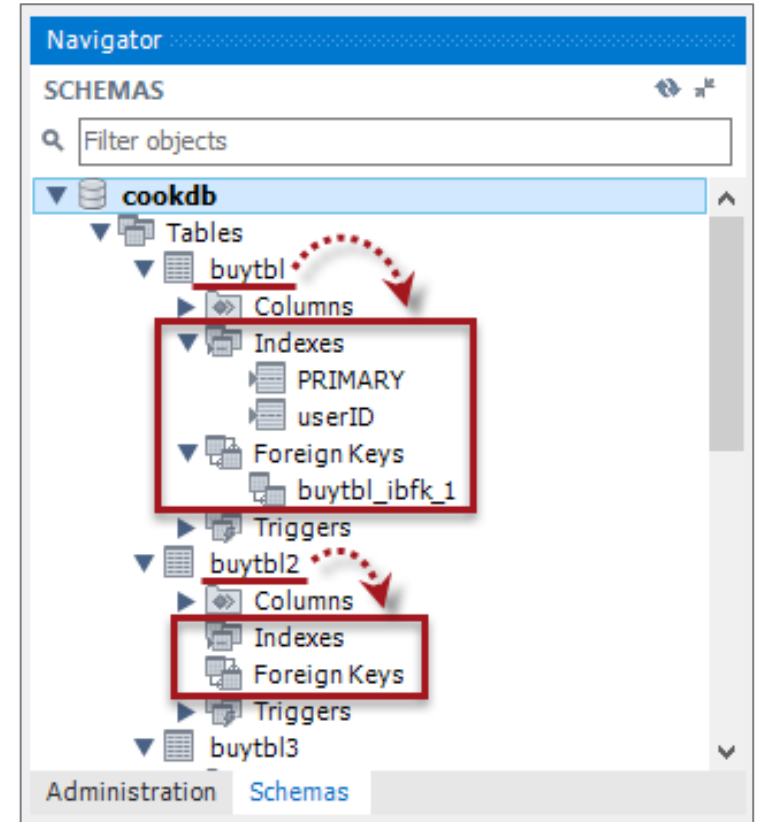
```
CREATE TABLE buyTBL3 (SELECT userID, prodName FROM buyTBL);  
SELECT * FROM buyTBL3;
```

The screenshot shows a database management interface. At the top, there's a 'Result Grid' section with a table containing 5 rows and 2 columns: 'userID' and 'prodName'. The rows are: KHD, 운동화; KHD, 노트북; PSH, 모니터; LHJ, 책; PSH, 운동화. Below this, there's a tab labeled 'buyTBL3 17 x' with a 'Read Only' status. The bottom section is titled 'Output' and shows 'Action Output' with a table of execution logs.

#	Time	Action	Message	Duration / Fetch
6	17:59:44	CREATE TABLE buyTBL3 (SELECT userID, prodNa...	12 row(s) affected Records: 12 Duplicates: 0 Wami...	0.078 sec
7	17:59:44	SELECT * FROM buyTBL3	12 row(s) returned	0.000 sec / 0.000 sec

2-9 CREATE TABLE ... SELECT 문

- 기본키와 외래키 등의 제약 조건은 복사되지 않음



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 two columns: 'userID' and 'amount'. The data rows are as follows:

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

Below the result grid, there's an 'Output' section with a dropdown menu set to 'Action Output'. It shows a log of database actions:

#	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

The '12 row(s) returned' message in the second row of the log is highlighted with a red rectangle.

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

■ 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);
```


3-4 WITH ROLLUP 절

- 분류(groupName)별 합계 및 그 총합을 구하기

```
SELECT num, groupName, SUM(price * amount) AS '비용'  
FROM buyTBL  
GROUP BY groupName, num  
WITH ROLLUP;
```

num	groupName	비용	
1	NULL	60	
10	NULL	60	
12	NULL	60	
NULL	NULL	180	소합계
7	서적	75	
8	서적	30	
11	서적	15	
NULL	서적	120	소합계
5	의류	150	
9	의류	50	
NULL	의류	200	소합계
2	전자	1000	
3	전자	200	
4	전자	1000	
6	전자	800	
NULL	전자	3000	소합계
NULL	NULL	3500	총합계

- 소합계와 총합만 필요하다면 num 뺀

```
SELECT groupName, SUM(price * amount) AS '비용'  
FROM buyTBL  
GROUP BY groupName  
WITH ROLLUP;
```

groupName	비용	
NULL	180	
서적	120	
의류	200	
전자	3000	
NULL	3500	총합계



Thank You
