

📌 DML (Data Manipulation Language)

정의: 테이블에 들어있는 데이터를 조회·추가·수정·삭제하는 언어.

대표 명령어:

SELECT (조회)

INSERT (추가)

UPDATE (수정)

DELETE (삭제)

특징: 가장 자주 쓰이며, 실제로 데이터를 조작하는 명령어.

📌 DDL (Data Definition Language)

정의: 데이터베이스의 구조(스키마)를 정의하거나 변경하는 언어.

대표 명령어:

CREATE (테이블, 데이터베이스 생성)

ALTER (구조 변경)

DROP (삭제)

특징: 데이터 자체가 아니라 **데이터를 담는 그릇(테이블, 스키마)**을 다룸.

📌 DCL (Data Control Language)

정의: 데이터베이스의 권한과 보안을 제어하는 언어.

대표 명령어:

GRANT (권한 부여)

REVOKE (권한 회수)

특징: 사용자 접근 권한을 관리.

SQL 언어 구성

DDL = 구조 정의

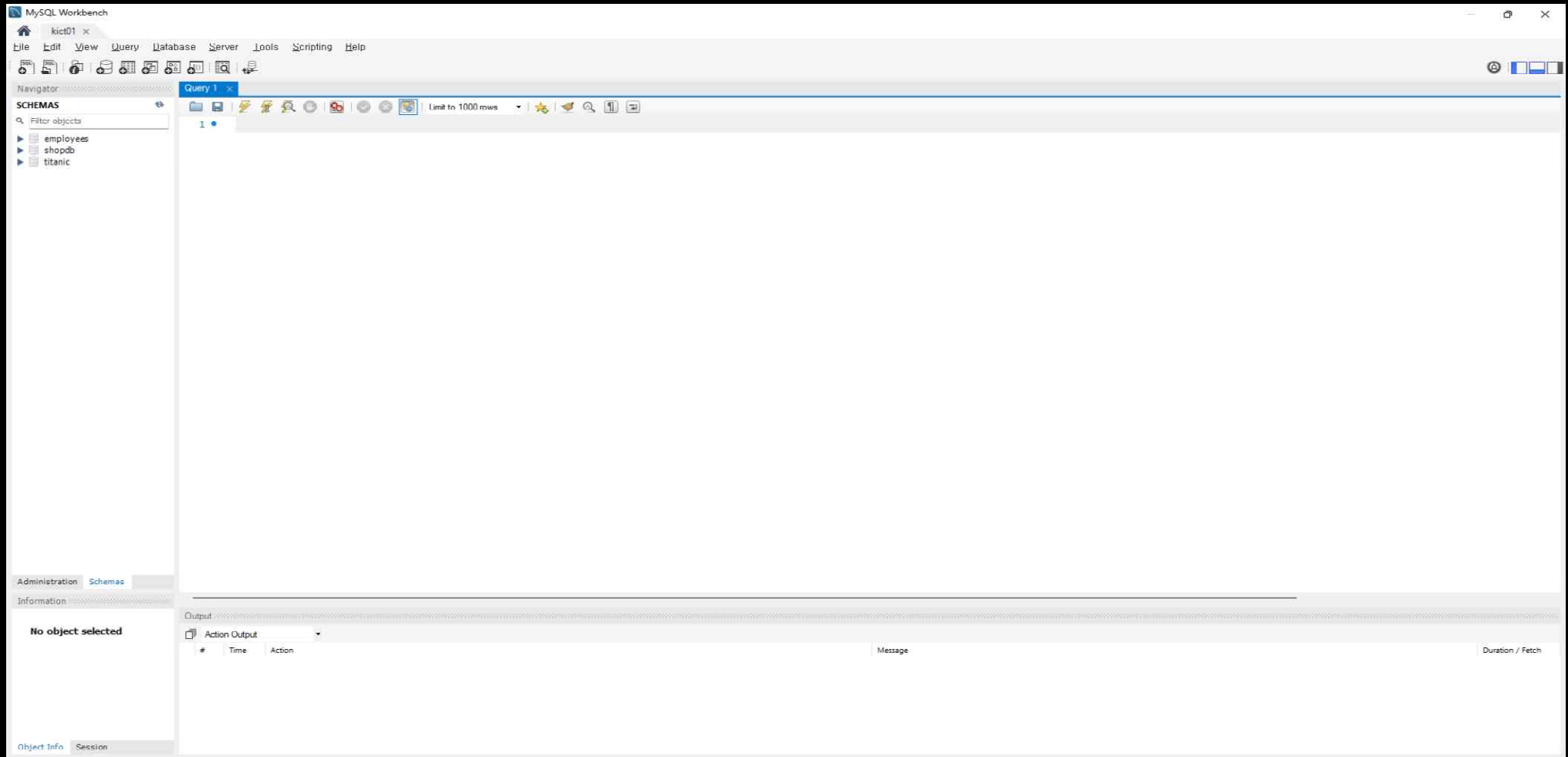
DML = 데이터 조작

DCL = 권한 제어

3. 보고 싶은 데이터 꺼내오기

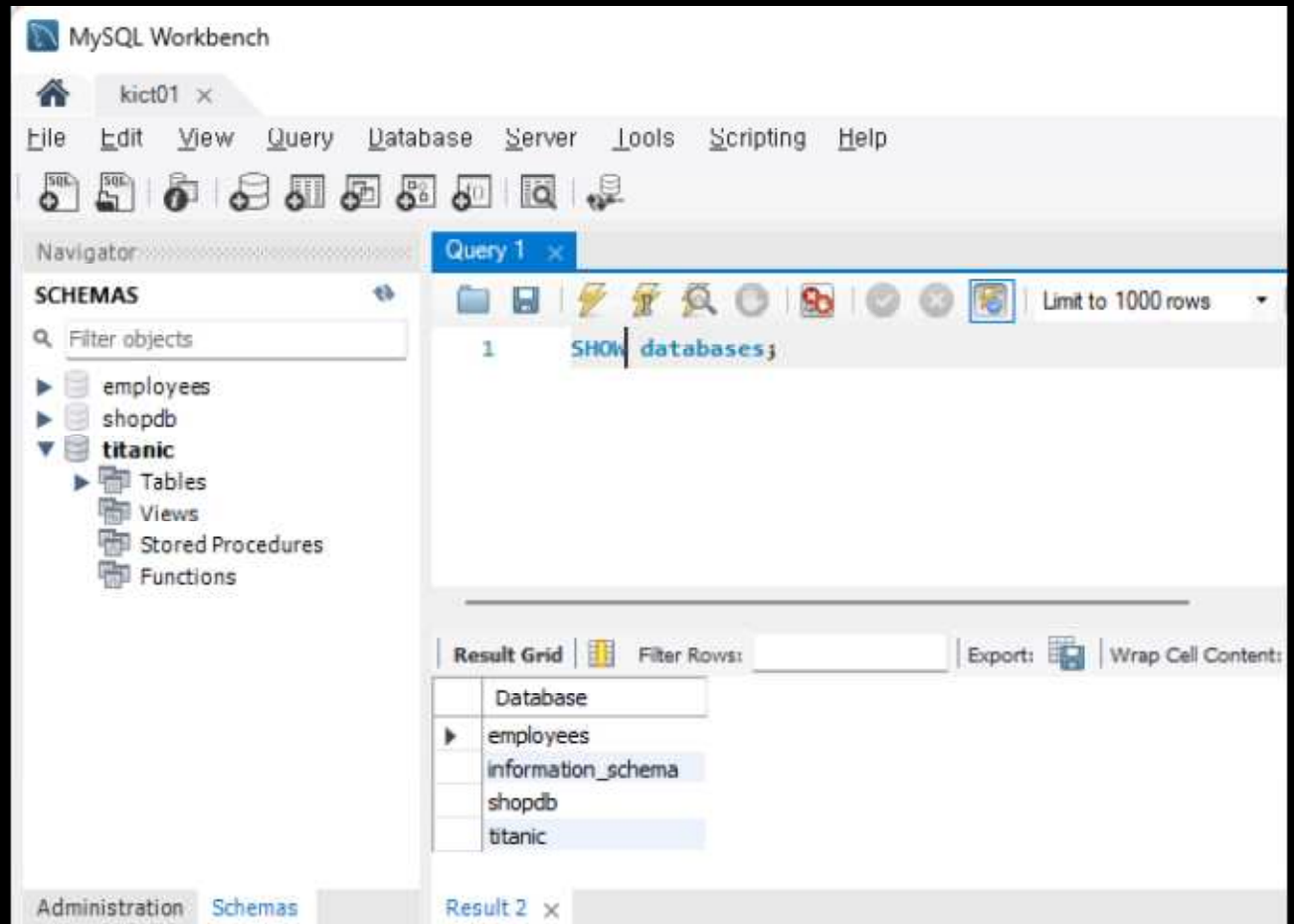
SHOW, USE, SELECT, FROM, LIMIT

Workbench의 Query창에 SQL명령 입력



SHOW databases- DATABASE 목록 표시

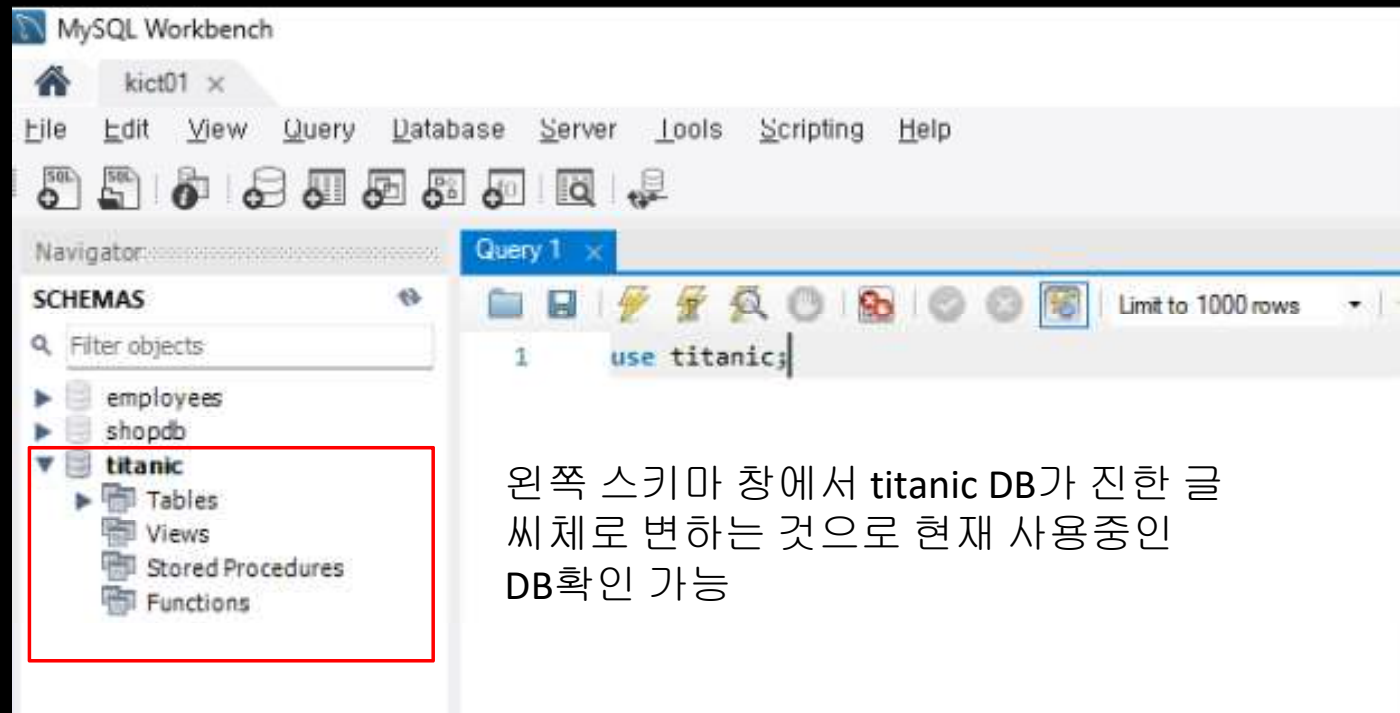
SHOW databases;
실행은 Ctrl + Enter



USE - 사용하고자 하는 DB 선택

USE 데이터베이스명; - 엑셀 파일을 여는 것과 같은 작업
실행은 Ctrl + Enter

USE titanic;



SELECT FROM - 테이블의 자료 검색하기

SELECT * FROM 테이블명; - 테이블의 모든 컬럼 조회

SELECT 컬럼명1, 컬럼명2 ... FROM 테이블명; - 원하는 컬럼만 조회

The image displays two side-by-side screenshots of the MySQL Workbench interface, demonstrating the execution of SQL queries and the resulting data.

Left Screenshot: The query editor shows the query `SELECT * FROM p_info;`. The result grid displays the following data:

PassengerId	Name	Sex	Age	SibSp	Parch
1	Braund, Mr. Owen Harris	male	22	1	0
2	Cummings, Mrs. John Bradley (Florence Briggs Th...	female	38	1	0
3	Heikkinen, Miss. Laina	female	26	0	0
4	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0
5	Allen, Mr. William Henry	male	35	0	0

The bottom panel shows the Action Output with the following log:

#	Time	Action	Message	Duration / Fetch
1	21:00:35	use titanic	0 row(s) affected	0.015 sec
2	21:04:09	show databases	4 row(s) returned	0.015 sec / 0.000 sec
3	21:04:20	SHOW databases	4 row(s) returned	0.000 sec / 0.000 sec
4	21:07:13	SHOW TABLES	3 row(s) returned	0.000 sec / 0.000 sec
5	21:31:10	SELECT * FROM p_info LIMIT 0, 1000	891 row(s) returned	0.015 sec / 0.000 sec

Right Screenshot: The query editor shows the query `SELECT Name, Sex, Age FROM p_info;`. The result grid displays the following data:

Name	Sex	Age
Braund, Mr. Owen Harris	male	22
Cummings, Mrs. John Bradley (Florence Briggs Th...	female	38
Heikkinen, Miss. Laina	female	26
Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35
Allen, Mr. William Henry	male	35

The bottom panel shows the Action Output with the following log:

#	Time	Action	Message	Duration / Fetch
2	21:04:09	show databases	4 row(s) returned	0.015 sec / 0.000 sec
3	21:04:20	SHOW databases	4 row(s) returned	0.000 sec / 0.000 sec
4	21:07:13	SHOW TABLES	3 row(s) returned	0.000 sec / 0.000 sec
5	21:31:10	SELECT * FROM p_info LIMIT 0, 1000	891 row(s) returned	0.015 sec / 0.000 sec
6	21:33:47	SELECT Name, Sex, Age FROM p_info LIMIT 0, 1000	891 row(s) returned	0.000 sec / 0.000 sec

SELECT FROM LIMIT- 검색 결과 제한

SELECT * FROM 테이블명 LIMIT 표시 개수

테이블의 모든 컬럼 조회
결과의 표시 개수를 10개
로 제한

SELECT * FROM p_info
LIMIT 10;

The screenshot shows the MySQL Workbench interface. The 'Navigator' pane on the left displays the 'titanic' database schema with tables 'p_info', 'survived', and 't_info'. The 'Query' editor shows the SQL statement: `SELECT * FROM p_info LIMIT 10;`. The 'Result Grid' pane displays the first 10 rows of the 'p_info' table. The 'Output' pane shows the execution log with the following entries:

#	Time	Action	Message	Duration / Fetch
3	21:04:20	SHOW databases	4 row(s) returned	0.000 sec / 0.000 sec
4	21:07:13	SHOW TABLES	3 row(s) returned	0.000 sec / 0.000 sec
5	21:31:10	SELECT * FROM p_info LIMIT 0, 1000	891 row(s) returned	0.015 sec / 0.000 sec
6	21:33:47	SELECT Name, Sex, Age FROM p_info LIMIT 0, 1000	891 row(s) returned	0.000 sec / 0.000 sec
7	21:45:23	SELECT * FROM p_info LIMIT 10	10 row(s) returned	0.000 sec / 0.000 sec

4. 조건에 맞는 데이터 검색하기

WHERE, 비교 연산자와 논리 연산자

WHERE 원하는 조건에 맞는 결과 조회

SELECT * FROM 테이블명 WHERE 조건;

P_info 테이블에서 나
이가 30살 이상인 사
람만 조회

SELECT * FROM P_info
where Age >= 30;

The screenshot shows the MySQL Workbench interface. The 'Navigator' pane on the left displays the 'titanic' database schema with tables 'p_info', 'survived', and 't_info'. The 'Query 1' editor contains the SQL query: `SELECT * FROM P_info where Age >= 30;`. The 'Result Grid' shows the following data:

PassengerId	Name	Sex	Age	SibSp	Parch
2	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38	1	0
4	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0
5	Allen, Mr. William Henry	male	35	0	0
7	McCarthy, Mr. Timothy J	male	54	0	0
12	Bonnell, Miss. Elizabeth	female	58	0	0

The 'Output' pane at the bottom shows the execution log with the following entries:

#	Time	Action	Message
8	22:58:38	SELECT * FROM P_info where 'Age' >= 30 LIMIT 0, 1000	Error Code: 1054. Un
9	22:58:46	SELECT * FROM P_info where Age >= 30 LIMIT 0, 1000	330 row(s) returned
10	22:59:21	SELECT * FROM P_info where Age >= 30 and Sex = 'male' LIMIT ...	Error Code: 1054. Un
11	22:59:28	SELECT * FROM P_info where Age >= 30 and Sex = 'male' LIMIT ...	216 row(s) returned
12	22:59:49	SELECT * FROM P_info where Age >= 30 LIMIT 0, 1000	330 row(s) returned

WHERE 원하는 조건에 맞는 결과 조회

SELECT * FROM 테이블명 WHERE 조건;

p_info 테이블에서 나이가
30살 이상인 남성만 조회

SELECT * FROM p_info
where Age >= 30 and Sex =
'male';

The screenshot shows the MySQL Workbench interface. The 'Navigator' pane on the left displays the 'titanic' database schema with tables 'p_info', 'survived', and 't_info'. The 'Query' pane shows the following SQL query:

```
1 SELECT * FROM P_info where Age >= 30 and Sex = 'male';
```

The 'Result Grid' pane displays the results of the query, showing 5 rows of data:

PassengerId	Name	Sex	Age	SibSp	Parch
5	Allen, Mr. William Henry	male	35	0	0
7	McCarthy, Mr. Timothy J	male	54	0	0
14	Andersson, Mr. Anders Johan	male	39	1	5
21	Fynney, Mr. Joseph J	male	35	0	0
22	Beesley, Mr. Lawrence	male	34	0	0

The 'Output' pane shows the execution log with the following entries:

#	Time	Action	Message
9	22:58:46	SELECT * FROM P_info where Age >= 30 LIMIT 0, 1000	330 row(s) returned
10	22:59:21	SELECT * FROM P_info where Age >= 30 and Sex = 'male' LIMIT ...	Error Code: 1054. Unk...
11	22:59:28	SELECT * FROM P_info where Age >= 30 and Sex = 'male' LIMIT ...	216 row(s) returned
12	22:59:49	SELECT * FROM P_info where Age >= 30 LIMIT 0, 1000	330 row(s) returned
13	23:21:46	SELECT * FROM P_info where Age >= 30 and Sex = 'male' LIMIT ...	216 row(s) returned

비교연산자

의미	연산자
같다	=
같지않다	!= , <>
오른쪽보다 왼쪽이 크다(초과)	>
오른쪽보다 왼쪽이 크다(이상)	>=
오른쪽보다 왼쪽이 작다(미만)	<
오른쪽보다 왼쪽이 작다(이하)	<=

논리연산자

의미	연산자
그리고	AND, &&
또는	OR,

WHERE 비교, 논리연산자로 조회하기

p_info 테이블에서 20살 이상 50세 미만의 여성을 조회하시오.

p_info 테이블에서 SibSp와 Parch가 1이상인 사람을 조회하시오.

t_info 테이블에서 Pclass가 1인 승객을 조회하시오.

t_info 테이블에서 Pclass가 2인 또는 Fare가 50 초과인 승객을 조회하시오.

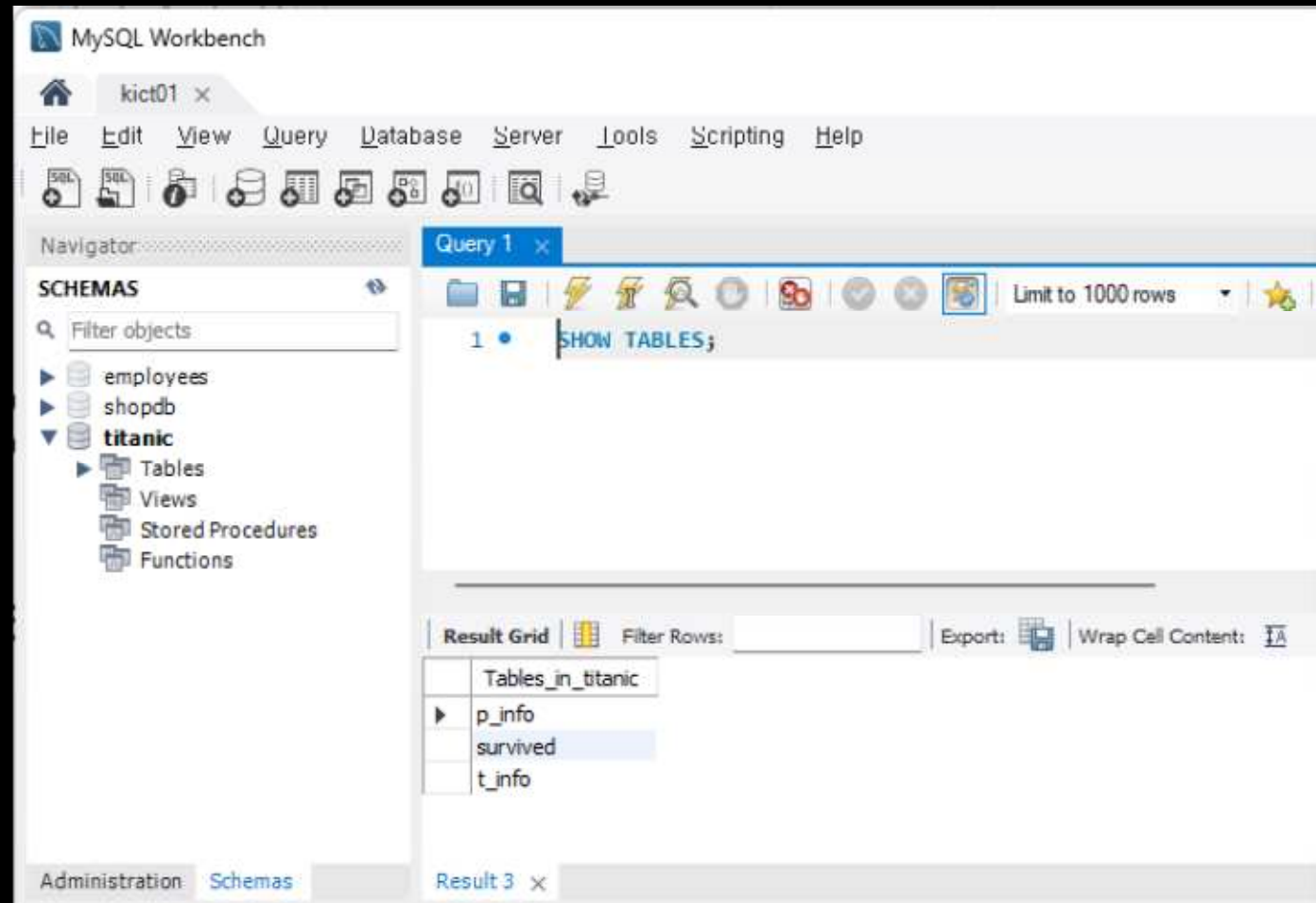
survived 테이블에서 Survived가 1인 승객을 조회하시오.

5. 조건에 맞는 데이터 검색하기 2

IN, LIKE, BETWEEN, IS NULL

SHOW tables - TABLE 목록 표시

SHOW tables;



기타연산자

의미	연산자
IN 안에 있는 값이 있는 경우	IN(값1, 값2 ...)
NOT IN 안에 있는 값이 없는 경우	NOT IN(값1, 값2 ...)
값이 처음, 끝, 어디든지 포함된 경우	LIKE('값%'), LIKE('%값'), LIKE('%값%')
값이 처음, 끝, 어디든지 포함되지 않은 경우	NOT LIKE('값%'), NOT LIKE('%값'), NOT LIKE('%값%')
a 이상 b 이하의 값	BETWEEN a AND b
NULL 인 경우	IS NULL
NULL 이 아닌 경우	IS NOT NULL

기타 연산자 in (찾을 값)

SELECT * FROM 테이블명 WHERE 컬럼명 IN (찾을 값);

SibSp 컬럼의 값이 1, 2, 3 인
경우의 행 찾기

SELECT * FROM p_info
WHERE SibSp IN (1,2,3);

OR로 조건을 묶은 것과 같음
SELECT * FROM p_info where
SibSp = 1 OR SibSp = 2 OR
SibSp = 3; 과 동일

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with 'titanic' selected, showing tables 'p_info', 'survived', and 't_info'. The main query editor shows two queries:

```
1 • SELECT * FROM p_info;  
2 • SELECT * FROM p_info WHERE SibSp IN (1,2,3);
```

The 'Result Grid' shows the results of the second query, displaying columns: PassengerId, Name, Sex, Age, SibSp, and Parch. The results are as follows:

PassengerId	Name	Sex	Age	SibSp	Parch
1	Braund, Mr. Owen Harris	male	22	1	0
2	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38	1	0
4	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0
8	Palsson, Master. Gosta Leonard	male	2	3	1
10	Nasser, Mrs. Nicholas (Adele Achem)	female	14	1	0
11	Sandstrom, Miss. Marguerite Rut	female	4	1	1

The bottom 'Action Output' pane shows the execution log:

#	Time	Action	Message
1	13:57:50	use titanic	0 row(s) affected
2	13:58:35	SELECT * FROM p_info LIMIT 0, 1000	891 row(s) returned
3	13:59:23	SELECT * FROM p_info where SibSp in (1,2,3) LIMIT 0, 1000	253 row(s) returned

기타 연산자 **not in** (찾을 값)

SELECT * FROM 테이블명 WHERE 컬럼명 NOT IN (찾을 값);

SibSp 컬럼의 값이 0, 1, 2, 3
이 아닌 경우의 행 찾기
**SELECT * FROM p_info
WHERE SibSp NOT IN
(1,2,3);**

AND로 조건을 묶은 것과
같음

**SELECT * FROM p_info
where SibSp != 0 AND SibSp
!= 1 AND SibSp != 2 AND
SibSp != 3; 과 동일**

The screenshot shows the MySQL Workbench interface. The 'Navigator' pane on the left shows the 'titanic' database with tables 'p_info', 'survived', and 't_info'. The 'Query 1' window shows the following SQL queries:

```
1 • SELECT * FROM p_info;  
2 • SELECT * FROM p_info WHERE SibSp NOT IN (0, 1, 2, 3);  
3 • SELECT * FROM p_info where SibSp != 0 AND SibSp != 1 AND SibSp != 2 AND SibSp != 3;
```

The 'Result Grid' shows the results of the first query, displaying columns: PassengerId, Name, Sex, Age, SibSp, and Parch. The results are as follows:

PassengerId	Name	Sex	Age	SibSp	Parch
17	Rice, Master. Eugene	male	2	4	1
51	Panula, Master. Juha Nilo	male	7	4	1
60	Goodwin, Master. William Frederick	male	11	5	2
69	Andersson, Miss. Erna Alexandra	female	17	4	2
72	Goodwin, Miss. Lillian Amy	female	16	5	2
120	Andersson, Miss. Ellis Anna Maria	female	2	4	2
160	Spang, Master. Thomas Henry	male	NULL	0	2

The 'Output' pane at the bottom shows the execution log with the following actions and messages:

#	Time	Action	Message
4	14:07:11	SELECT * FROM p_info where SibSp = 1 OR SibSp = 2 or SibSp = ...	253 row(s) returned
5	14:09:13	SELECT * FROM p_info WHERE SibSp NOT IN (1,2,3) LIMIT 0, 1...	638 row(s) returned
6	14:09:26	SELECT * FROM p_info WHERE SibSp NOT IN (0, 1,2,3) LIMIT 0...	30 row(s) returned
7	14:10:10	SELECT * FROM p_info where SibSp != 0 OR SibSp != 1 OR SibS...	891 row(s) returned
8	14:10:39	SELECT * FROM p_info where SibSp != 0 AND SibSp != 1 AND Si...	30 row(s) returned

기타 연산자 LIKE (찾을 값) - 문자열 검색

SELECT * FROM 테이블명 WHERE 컬럼명 LIKE (찾을 값);

의미	연산자
정확히 일치	LIKE('값')
자료 앞에 포함	LIKE('값%')
자료 끝에 포함	LIKE('%값')
자료 중간에 포함	LIKE('%값%')

기타 연산자 LIKE (찾을 값)

SELECT * FROM 테이블명 WHERE 컬럼명 LIKE (찾을 값);

Name 컬럼의 값에서 Braund라는 이름을
찾는 사람 찾기

SELECT * FROM p_info WHERE Name LIKE
'Braund'; => 정확히 일치하지 않기 때문에
결과 조회 불가.

문자열 안에 단어가 포함된 것을 찾기 위
해서는 %를 써서 단어가 있는 위치를 지
정해주어야 함.

문자열 중간이라면 %찾는단어%

The top screenshot shows a query: `SELECT * FROM p_info WHERE Name LIKE 'Braund';`. The result grid is empty, and the text "결과가 조회되지 않음" (Results not found) is displayed in red.

The bottom screenshot shows a query: `SELECT * FROM p_info WHERE Name LIKE 'Braund%';`. The result grid contains two rows:

PassengerId	Name	Sex	Age	SibSp	Parch
1	Braund, Mr. Owen Harris	male	22	1	0
478	Braund, Mr. Lewis Richard	male	29	1	0

The text "결과가 조회 됨" (Results found) is displayed in red.

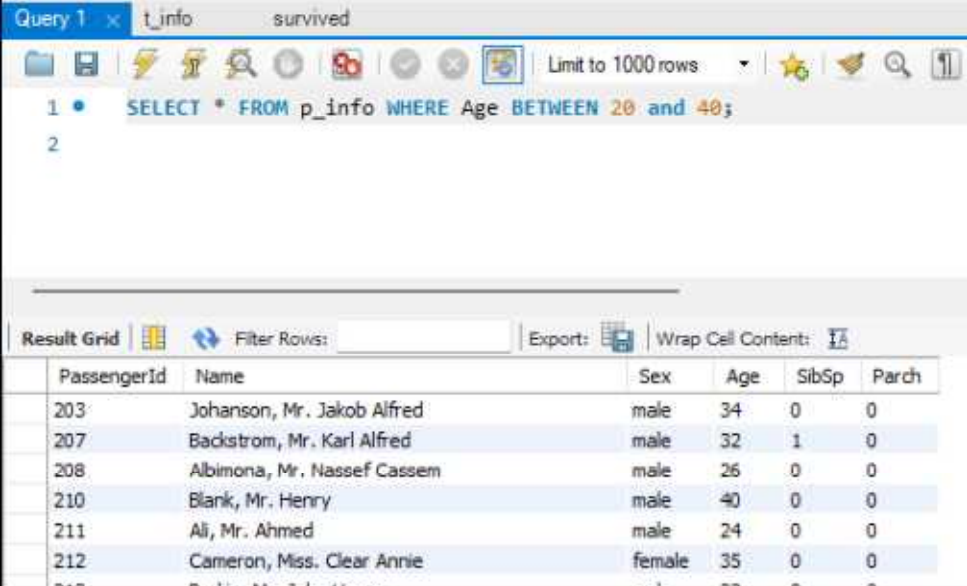
기타 연산자 BETWEEN A AND B (범위)

SELECT * FROM 테이블명 WHERE 컬럼명 BETWEEN A AND B;

Age 컬럼의 값이 20 이상 40 이하인 값 찾기

SELECT * FROM p_info WHERE Age BETWEEN 20 AND 40;

SELECT * FROM p_info WHERE Age >= 20 AND Age <= 40; 와 같은 결과



The screenshot shows a SQL query editor window titled 'Query 1' with a tab 't_info' and a status bar 'survived'. The query is: `SELECT * FROM p_info WHERE Age BETWEEN 20 and 40;`. The results are displayed in a table with columns: PassengerId, Name, Sex, Age, SibSp, and Parch. The results show 6 rows of data.

PassengerId	Name	Sex	Age	SibSp	Parch
203	Johanson, Mr. Jakob Alfred	male	34	0	0
207	Backstrom, Mr. Karl Alfred	male	32	1	0
208	Albinona, Mr. Nassef Cassem	male	26	0	0
210	Blank, Mr. Henry	male	40	0	0
211	Ali, Mr. Ahmed	male	24	0	0
212	Cameron, Miss. Clear Annie	female	35	0	0

WHERE 기타연산자로 조회하기

t_info 테이블에서 Fare가 100 이상 1000이하인 승객을 조회하시오.

t_info 테이블에서 Ticket이 PC로 시작하고 Embarked가 C 혹은 S인 승객을 조회하시오.

t_info 테이블에서 Pclass가 1 혹은 2인 승객을 조회하시오.

t_info 테이블에서 Cabin에 숫자 59가 포함된 승객을 조회하시오.

p_info 테이블에서 Age가 NULL이 아니면서 이름에 James가 포함된 40세 이상의 남성을 조회하시오.

기타 연산자 IS NULL/IS NOT NULL (결측)

SELECT * FROM 테이블명 WHERE 컬럼명 IS NULL;

Age 컬럼의 값이 NULL(값이 없음)인 행 찾기
SELECT * FROM p_info WHERE Age IS NULL;

Age 컬럼과 SibSp 컬럼 값이 NULL이 아닌 행 찾기

SELECT * FROM p_info WHERE Age IS NOT NULL AND SibSp IS NOT NULL;

The screenshot shows a SQL query editor with the following query:

```
1 SELECT * FROM p_info WHERE Age is null;
2
```

The query is executed, and the results are displayed in a table with the following columns: PassengerId, Name, Sex, Age, SibSp, and Parch. The Age column contains NULL values for the first six rows shown.

PassengerId	Name	Sex	Age	SibSp	Parch
6	Moran, Mr. James	male	NULL	0	0
18	Williams, Mr. Charles Eugene	male	NULL	0	0
20	Massemani, Mrs. Fatima	female	NULL	0	0
27	Emir, Mr. Farred Chehab	male	NULL	0	0
29	O'Dwyer, Miss. Ellen "Nellie"	female	NULL	0	0
30	Todoroff, Mr. Lalio	male	NULL	0	0

The output section shows the action log with the following entries:

#	Time	Action	Message
21	15:26:16	SELECT * FROM p_info WHERE Name LIKE 'Braund%' LIMIT 0, 1...	2 row(s) returned
22	15:36:01	SELECT * FROM p_info WHERE Age BETWEEN 20 and 40 LIM...	400 row(s) returned
23	15:38:23	SELECT * FROM p_info WHERE Age >= 20 and Age <= 40 LIMIT...	400 row(s) returned
24	15:44:04	SELECT * FROM p_info WHERE Age isnull	Error Code: 1064. Yo
25	15:44:22	SELECT * FROM p_info WHERE Age is null LIMIT 0, 1000	177 row(s) returned

6. 데이터 순서 정렬하기

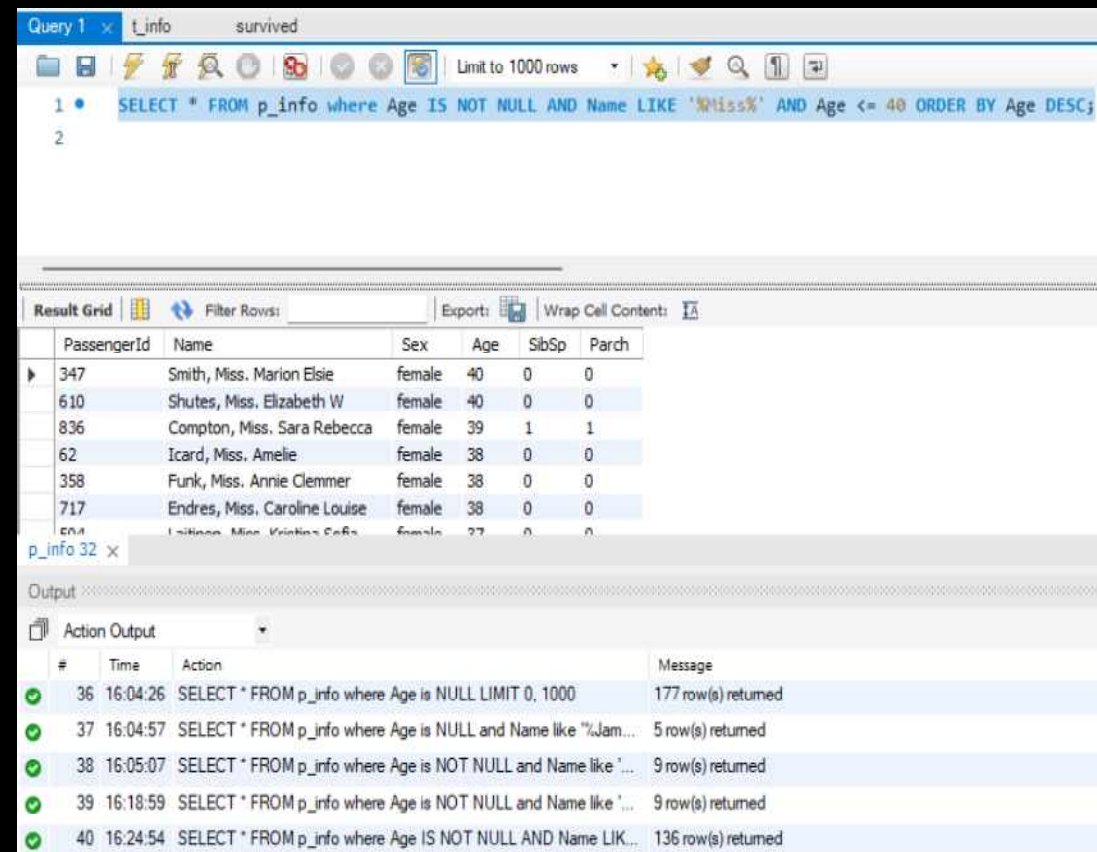
ORDER BY, GROUP BY, HAVING

ORDER BY - 조회 된 결과를 정렬

SELECT * FROM 테이블명 WHERE 컬럼명 ORDER BY 기준컬럼 ;
ASC 오름차순, DESC 내림차순

p_info 테이블에서 Age가 NULL이 아니면서 이름에 Miss가 포함된 40세 이하의 여성을 조회하고 나이를 기준으로 내림차순 정렬 하시오.

```
SELECT * FROM p_info where Age IS NOT NULL AND Name LIKE '%Miss%' AND Age <= 40 ORDER BY Age DESC;
```



The screenshot shows a SQL query execution interface. The query is: `SELECT * FROM p_info where Age IS NOT NULL AND Name LIKE '%Miss%' AND Age <= 40 ORDER BY Age DESC;`. The results are displayed in a table with columns: PassengerId, Name, Sex, Age, SibSp, and Parch. The results are ordered by Age in descending order.

PassengerId	Name	Sex	Age	SibSp	Parch
347	Smith, Miss. Marion Elsie	female	40	0	0
610	Shutes, Miss. Elizabeth W	female	40	0	0
836	Compton, Miss. Sara Rebecca	female	39	1	1
62	Icard, Miss. Amelie	female	38	0	0
358	Funk, Miss. Annie Clemmer	female	38	0	0
717	Endres, Miss. Caroline Louise	female	38	0	0
504	Lehman, Miss. Virginia Soth	female	27	0	0

The interface also shows an 'Output' section with a table of actions and messages:

#	Time	Action	Message
36	16:04:26	SELECT * FROM p_info where Age is NULL LIMIT 0, 1000	177 row(s) returned
37	16:04:57	SELECT * FROM p_info where Age is NULL and Name like '%Jam...'	5 row(s) returned
38	16:05:07	SELECT * FROM p_info where Age is NOT NULL and Name like '%Miss%'	9 row(s) returned
39	16:18:59	SELECT * FROM p_info where Age is NOT NULL and Name like '%Miss%' AND Age <= 40	9 row(s) returned
40	16:24:54	SELECT * FROM p_info where Age IS NOT NULL AND Name LIKE '%Miss%' AND Age <= 40 ORDER BY Age DESC;	136 row(s) returned

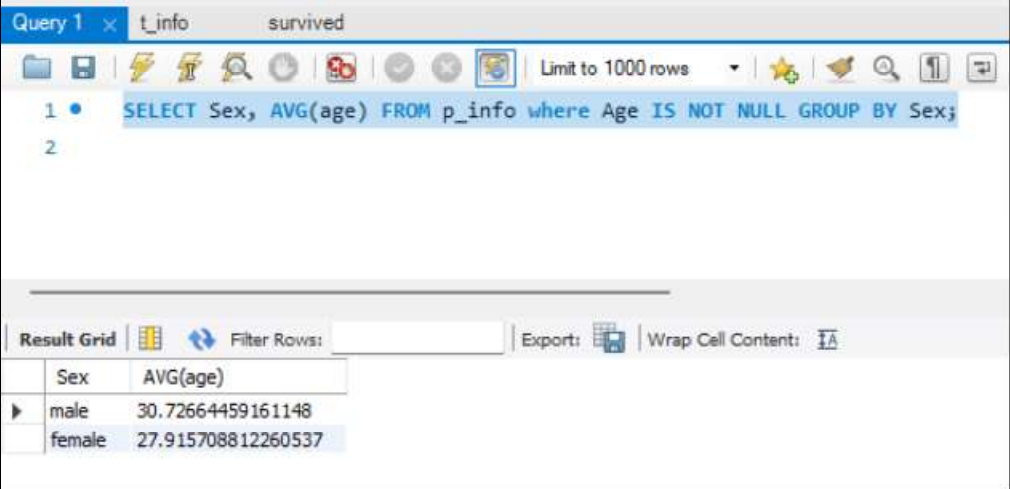
GROUP BY - 특정 컬럼 값을 기준으로 그룹 연산

SELECT 기준 컬럼명, 그룹연산함수 FROM 테이블명 WHERE 컬럼명
GROUP BY 기준컬럼 ;

함수명	기능
AVG()	평균
MIN()	최소값
MAX()	최대값
COUNT()	행 개수
COUNT(DISTINCT)	중복 값이 없는 행 개수

p_info 테이블에서 나이가 Null이 아닌 행의 성별
별 나이 평균을 구하시오.

```
SELECT Sex, AVG(age) FROM p_info where Age IS  
NOT NULL GROUP BY Sex;
```



The screenshot shows a SQL query editor with the following query: `SELECT Sex, AVG(age) FROM p_info where Age IS NOT NULL GROUP BY Sex;` The results are displayed in a table with two columns: Sex and AVG(age). The results are as follows:

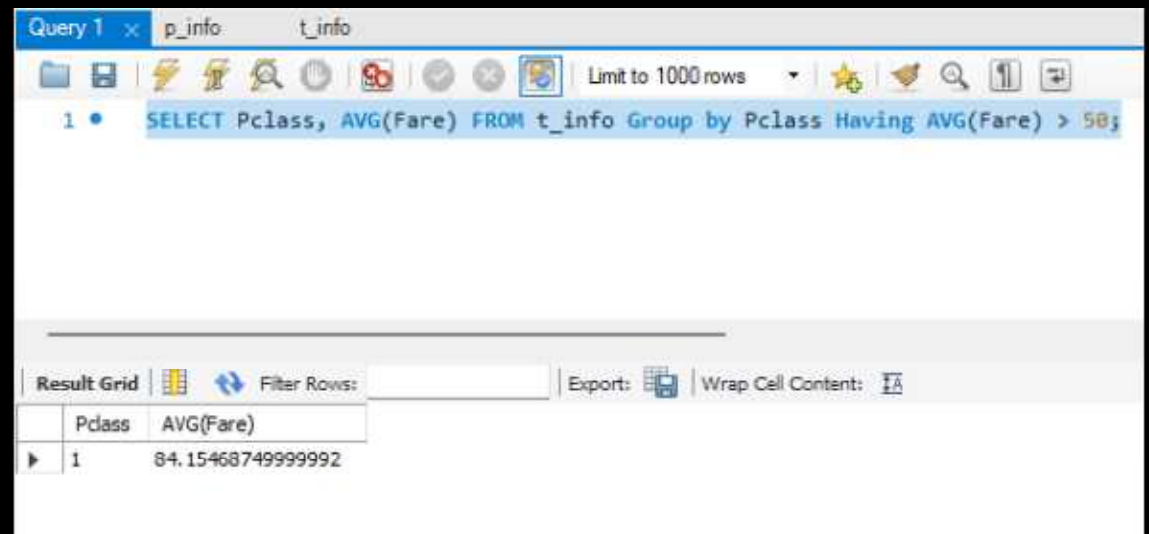
Sex	AVG(age)
male	30.72664459161148
female	27.915708812260537

GROUP BY HAVING - 특정 컬럼 그룹 연산 결과에서 원하는 결과만 다시 추출할 때

SELECT 기준 컬럼명, 그룹연산함수1 **FROM** 테이블명 **GROUP BY** 기준컬럼 **HAVING** 조건;

t_info 테이블에서 Pclass별 Fare 가격 평균을 구하고 그 중 가격 평균이 50을 초과하는 결과만 조회.

```
SELECT Pclass, AVG(Fare) FROM t_info  
Group by Pclass Having AVG(Fare) >  
50;
```



The screenshot shows a SQL query editor window with a query titled 'Query 1'. The query is: `SELECT Pclass, AVG(Fare) FROM t_info Group by Pclass Having AVG(Fare) > 50;`. The results are displayed in a table with two columns: 'Pclass' and 'AVG(Fare)'. The first row shows '1' for Pclass and '84.15468749999992' for the average fare.

	Pclass	AVG(Fare)
1	1	84.15468749999992

7. 여러 곳에 분산된 데이터를 모아서 가져오기

JOIN(INNER, LEFT, RIGHT, OUTER)

JOIN - 2개 혹은 2개 이상의 테이블을 합쳐서 출력

SELECT * FROM 테이블1명 INNER JOIN 테이블2명 ON 테이블1명.
.기준컬럼명 = 테이블2명.기준컬럼명;
테이블명 as 약칭 - 긴 테이블명을 짧게 줄임

의미	연산자
왼쪽과 오른쪽 테이블의 기준 컬럼에서 서로 일치하는 자료만 합침(교집합)	INNER JOIN
왼쪽 테이블의 기준 컬럼에 있는 자료에 해당하는 값만 오른쪽 테이블에서 합침	LEFT JOIN
오른쪽 테이블의 기준 컬럼에 있는 자료에 해당하는 값만 왼쪽 테이블에서 합침	RIGHT JOIN
왼쪽과 오른쪽 테이블의 모든 자료를 합침	LEFT JOIN UNION ALL RIGHT JOIN

INNER JOIN - (교집합) 기준 컬럼을 비교해 양쪽에 데이터가 있는 행만 합쳐줌

p_info 테이블						survived 테이블	
PassengerId	Name	Sex	Age	SibSp	Parch	PassengerId	Survived
1	Braund, Mr. Owen Harris	male	22	1	0	2	0
2	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	4	1
3	Heikkinen, Miss. Laina	female	26	0	0	6	1
4	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	7	1
5	Allen, Mr. William Henry	male	35	0	0	8	0
INNER JOIN (교집합)						9	0
						10	0
						11	0
						12	1
						13	1
PassengerId를 기준으로 INNER JOIN 2와 4만 양쪽에 모두 존재하므로 PassengerId 2, 4만 합쳐짐							
p_info 테이블						survived 테이블	
PassengerId	Name	Sex	Age	SibSp	Parch	Survived	
2	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	1	
4	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	1	

INNER JOIN - (교집합)

SELECT * FROM passenger AS p INNER JOIN ticket AS t ON p.PassengerId = t.PassengerId;

passenger테이블 623행
ticket테이블 445행

INNER JOIN 후 317행

The screenshot shows a database query window with the following SQL query:

```
1 SELECT * FROM passenger as p INNER JOIN ticket as t ON p.PassengerId = t.PassengerId;
```

The query results are displayed in a table with the following columns: Name, Sex, Age, SibSp, Parch, PassengerId, Ticket, Pclass, Fare, Cabin, Embarked.

Name	Sex	Age	SibSp	Parch	PassengerId	Ticket	Pclass	Fare	Cabin	Embarked
Arbines, Mr. William	male	19	0	0	192	28424	2	13	NULL	S
Greenberg, Mr. Samuel	male	52	0	0	715	250647	2	13	NULL	S
Wheeler, Mr. Joseph Jr	male	17	1	1	533	2690	3	7.2292	NULL	C
Robins, Mrs. Alexander A (Grace Charity Laury)	female	47	1	0	133	A/5. 3337	3	14.5	NULL	S
White, Miss. Jessie Wills	female	NULL	0	0	597	248727	2	33	NULL	S

Below the query results, the 'Action Output' section shows the execution progress:

#	Time	Action	Message	Duration / P
15	12:58:52	truncate ticket	0 row(s) affected	0.015 sec
16	13:01:15	SELECT * FROM titanic.passenger LIMIT 0, 1000	623 row(s) returned	0.000 sec /
17	13:01:20	SELECT * FROM titanic.surv LIMIT 0, 1000	446 row(s) returned	0.016 sec /
18	13:01:24	SELECT * FROM titanic.ticket LIMIT 0, 1000	445 row(s) returned	0.000 sec /
19	13:03:15	SELECT * FROM passenger as p INNER JOIN ticket as t ON p.Pa...	317 row(s) returned	0.000 sec /

LEFT JOIN - 왼쪽 테이블을 기준으로 합침

p_info 테이블						survived 테이블	
PassengerId	Name	Sex	Age	SibSp	Parch	PassengerId	Survived
1	Braund, Mr. Owen Harris	male	22	1	0	1	0
2	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	3	1
3	Heikkinen, Miss. Laina	female	26	0	0	5	1
4	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	6	1
5	Allen, Mr. William Henry	male	35	0	0	7	0
LEFT JOIN						8	0
						9	0
						10	0
						11	1
						12	1
PassengerId를 기준으로 LEFT JOIN							
p_info 테이블의 PassengerId가 1-5까지 있으므로 survived 테이블에서 PassengerId 1-5에 해당하는 데이터만 가져와 합침							
p_info 테이블						survived 테이블	
PassengerId	Name	Sex	Age	SibSp	Parch	Survived	
1	Braund, Mr. Owen Harris	male	22	1	0	0	
2	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	null	
3	Heikkinen, Miss. Laina	female	26	0	0	1	
4	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	null	
5	Allen, Mr. William Henry	male	35	0	0	1	

LEFT JOIN

SELECT * FROM passenger AS p LEFT JOIN ticket AS t ON p.PassengerId = t.PassengerId;

passenger테이블 623행
ticket테이블 445행

LEFT JOIN 후 623행

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

```
1 SELECT * FROM passenger as p LEFT JOIN ticket as t ON p.PassengerId = t.PassengerId;
```

The results window shows a table with 12 columns: PassengerId, Name, Sex, Age, SibSp, Parch, PassengerId, Ticket, Pclass, Fare, Cabin, and Embarked. The table contains 623 rows, with the first five rows displayed. The first row is for Stead, Mr. William Thomas, a male, age 62, with 0 siblings and 0 parents, and a null ticket. The second row is for Leitch, Miss. Jessie Wills, a female, age 32, with 0 siblings and 0 parents, and ticket 248727. The third row is for Harrison, Mr. William, a male, age 40, with 0 siblings and 0 parents, and ticket 112059. The fourth row is for Jenkin, Mr. Stephen Curnow, a male, age 32, with 0 siblings and 0 parents, and a null ticket. The fifth row is for Mellinger, Mrs. (Elizabeth Anne Maidment), a female, age 41, with 0 siblings and 1 parent, and a null ticket.

PassengerId	Name	Sex	Age	SibSp	Parch	PassengerId	Ticket	Pclass	Fare	Cabin	Embarked
1	Stead, Mr. William Thomas	male	62	0	0	NULL	NULL	NULL	NULL	NULL	NULL
2	Leitch, Miss. Jessie Wills	female	32	0	0	597	248727	2	33	NULL	S
3	Harrison, Mr. William	male	40	0	0	264	112059	1	0	B94	S
4	Jenkin, Mr. Stephen Curnow	male	32	0	0	NULL	NULL	NULL	NULL	NULL	NULL
5	Mellinger, Mrs. (Elizabeth Anne Maidment)	female	41	0	1	NULL	NULL	NULL	NULL	NULL	NULL

The results window also shows a log of the query execution, including the query text and the number of rows returned. The log shows that the query returned 623 rows.

#	Time	Action	Message	Duration / Fetch
16	13:01:15	SELECT * FROM titanic.passenger LIMIT 0, 1000	623 row(s) returned	0.000 sec / 0.000 sec
17	13:01:20	SELECT * FROM titanic.surv LIMIT 0, 1000	446 row(s) returned	0.016 sec / 0.000 sec
18	13:01:24	SELECT * FROM titanic.ticket LIMIT 0, 1000	445 row(s) returned	0.000 sec / 0.000 sec
19	13:03:15	SELECT * FROM passenger as p INNER JOIN ticket as t ON p.PassengerId = t.PassengerId	317 row(s) returned	0.000 sec / 0.000 sec
20	14:23:51	SELECT * FROM passenger as p LEFT JOIN ticket as t ON p.PassengerId = t.PassengerId	623 row(s) returned	0.016 sec / 0.000 sec

RIGHT JOIN - 오른쪽 테이블을 기준으로 합침

p_info 테이블						survived 테이블	
PassengerId	Name	Sex	Age	SibSp	Parch	PassengerId	Survived
1	Braund, Mr. Owen Harris	male	22	1	0	1	0
2	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	2	1
3	Heikkinen, Miss. Laina	female	26	0	0	3	1
4	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	4	1
5	Allen, Mr. William Henry	male	35	0	0	5	0
RIGHT JOIN						6	0
						7	0
						8	0
						9	1
						10	1
survived 테이블의 PassengerId가 1-10까지 있으므로 p_info 테이블에서 PassengerId 1-10까지 가져와야하지만							
1-5까지밖에 없으므로 1-5까지만 가져옴. 6-10까지는 데이터가 없으므로 null값으로 채워짐							
p_info 테이블						survived 테이블	
PassengerId	Name	Sex	Age	SibSp	Parch	Survived	
1	Braund, Mr. Owen Harris	male	22	1	0	0	
2	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	1	
3	Heikkinen, Miss. Laina	female	26	0	0	1	
4	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	1	
5	Allen, Mr. William Henry	male	35	0	0	0	
6	null	null	null	null	null	0	
7	null	null	null	null	null	0	
8	null	null	null	null	null	0	
9	null	null	null	null	null	1	
10	null	null	null	null	null	1	

RIGHT JOIN

SELECT * FROM passenger AS p RIGHT JOIN ticket AS t ON p.PassengerId = t.PassengerId;

passenger테이블 623행
ticket테이블 445행

RIGHT JOIN 후 445행

The screenshot displays a SQL query execution environment. At the top, a query window shows the SQL statement: `SELECT * FROM passenger as p RIGHT JOIN ticket as t ON p.PassengerId = t.PassengerId;`. Below the query, a 'Result Grid' shows the results of the query. The grid has 13 columns: PassengerId, Name, Sex, Age, SibSp, Parch, PassengerId, Ticket, Pclass, Fare, Cabin, and Embarked. The results show four rows where the passenger has a corresponding ticket, with the first column (PassengerId) being NULL. Below the result grid, an 'Output' section shows the 'Action Output' log, which includes the execution time and the number of rows returned for each step of the query execution.

PassengerId	Name	Sex	Age	SibSp	Parch	PassengerId	Ticket	Pclass	Fare	Cabin	Embarked
NULL	NULL	NULL	NULL	NULL	NULL	486	4133	3	25.4667	NULL	S
119	Baxter, Mr. Quigg Edmond	male	24	0	1	119	PC 17558	1	247.5208	B58 B60	C
NULL	NULL	NULL	NULL	NULL	NULL	836	PC 17756	1	83.1583	E49	C
528	Farthing, Mr. John	male	NULL	0	0	528	PC 17483	1	221.7792	C95	S
396	Johansson, Mr. Erik	male	22	0	0	396	350052	3	7.7958	NULL	S

#	Time	Action	Message	Duration / F
17	13:01:20	SELECT * FROM titanic.surv LIMIT 0, 1000	446 row(s) returned	0.016 sec /
18	13:01:24	SELECT * FROM titanic.ticket LIMIT 0, 1000	445 row(s) returned	0.000 sec /
19	13:03:15	SELECT * FROM passenger as p INNER JOIN ticket as t ON p.Pa...	317 row(s) returned	0.000 sec /
20	14:23:51	SELECT * FROM passenger as p LEFT JOIN ticket as t ON p.Pass...	623 row(s) returned	0.016 sec /
21	14:37:12	SELECT * FROM passenger as p RIGHT JOIN ticket as t ON p.Pa...	445 row(s) returned	0.016 sec /

FULL OUTER JOIN - 양쪽 테이블을 모두 합침

p_info 테이블						survived 테이블	
PassengerId	Name	Sex	Age	SibSp	Parch	PassengerId	Survived
1	Braund, Mr. Owen Harris	male	22	1	0	11	0
2	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	12	1
3	Heikkinen, Miss. Laina	female	26	0	0	13	1
4	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	14	1
5	Allen, Mr. William Henry	male	35	0	0	15	0
						16	0
						17	0
						18	0
						19	1
						20	1

FULL OUTER JOIN (합집합)

MySQL에는 FULL OUTER JOIN 명령이 없어 UNION ALL 이라는 명령으로 LEFT, RIGHT JOIN 결과를 합쳐야 함

p_info테이블의 PassengerId는 1-5 survived 테이블의 PssengerId는 11-20으로 일치되는 행이 없음

그러나 FULL OUTER JOIN은 합집합이므로 두 테이블의 모든 데이터가 합쳐짐

p_info 테이블						survived 테이블
PassengerId	Name	Sex	Age	SibSp	Parch	Survived
1	Braund, Mr. Owen Harris	male	22	1	0	null
2	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	null
3	Heikkinen, Miss. Laina	female	26	0	0	null
4	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	null
5	Allen, Mr. William Henry	male	35	0	0	null
11	null	null	null	null	null	0
12	null	null	null	null	null	1
13	null	null	null	null	null	1
14	null	null	null	null	null	1
15	null	null	null	null	null	0
16	null	null	null	null	null	0
17	null	null	null	null	null	0
18	null	null	null	null	null	0
19	null	null	null	null	null	1
20	null	null	null	null	null	1

FULL OUTER JOIN

- SELECT * FROM passenger AS p LEFT JOIN ticket AS t ON p.PassengerId = t.PassengerId
- UNION
- SELECT * FROM passenger AS p RIGHT JOIN ticket AS t ON p.PassengerId = t.PassengerId;

passenger테이블 623행
ticket테이블 445행

Full OUTER JOIN 후 1068행
중복제거 후 751행

The screenshot shows a SQL query editor with the following query:

```
1 SELECT * FROM passenger as p LEFT JOIN ticket as t ON p.PassengerId = t.PassengerId
2 UNION
3 SELECT * FROM passenger as p RIGHT JOIN ticket as t ON p.PassengerId = t.PassengerId;
```

Below the query, the 'Result Grid' shows the first five rows of the result set:

PassengerId	Name	Sex	Age	SibSp	Parch	PassengerId	Ticket	Pclass	Fare	Cabin	Emb
658	Bourke, Mrs. John (Catherine)	female	32	1	1	HULL	HULL	HULL	HULL	HULL	HULL
303	Johnson, Mr. William Cahoon Jr	male	19	0	0	HULL	HULL	HULL	HULL	HULL	HULL
787	Sjoblom, Miss. Anna Sofia	female	18	0	0	HULL	HULL	HULL	HULL	HULL	HULL
814	Andersson, Miss. Ebba Iris Alfrida	female	6	4	2	HULL	HULL	HULL	HULL	HULL	HULL
594	Bourke, Miss. Mary	female	HULL	0	2	HULL	HULL	HULL	HULL	HULL	HULL

Below the result grid, the 'Output' section shows the execution log:

#	Time	Action	Message	Duration / Fetc
20	14:23:51	SELECT * FROM passenger as p LEFT JOIN ticket as t ON p.Pass...	623 row(s) returned	0.016 sec / 0.
21	14:37:12	SELECT * FROM passenger as p RIGHT JOIN ticket as t ON p.Pa...	445 row(s) returned	0.016 sec / 0.
22	14:48:23	SELECT * FROM passenger as p LEFT JOIN ticket as t ON p.Pass...	Error Code: 1146. Table 'titanic.t' doesn't exist	0.016 sec
23	14:48:55	SELECT * FROM passenger as p LEFT JOIN ticket as t ON p.Pass...	1068 row(s) returned	0.016 sec / 0.
24	14:49:23	SELECT * FROM passenger as p LEFT JOIN ticket as t ON p.Pass...	751 row(s) returned	0.016 sec / 0.

조인 후 원하는 컬럼 검색하기

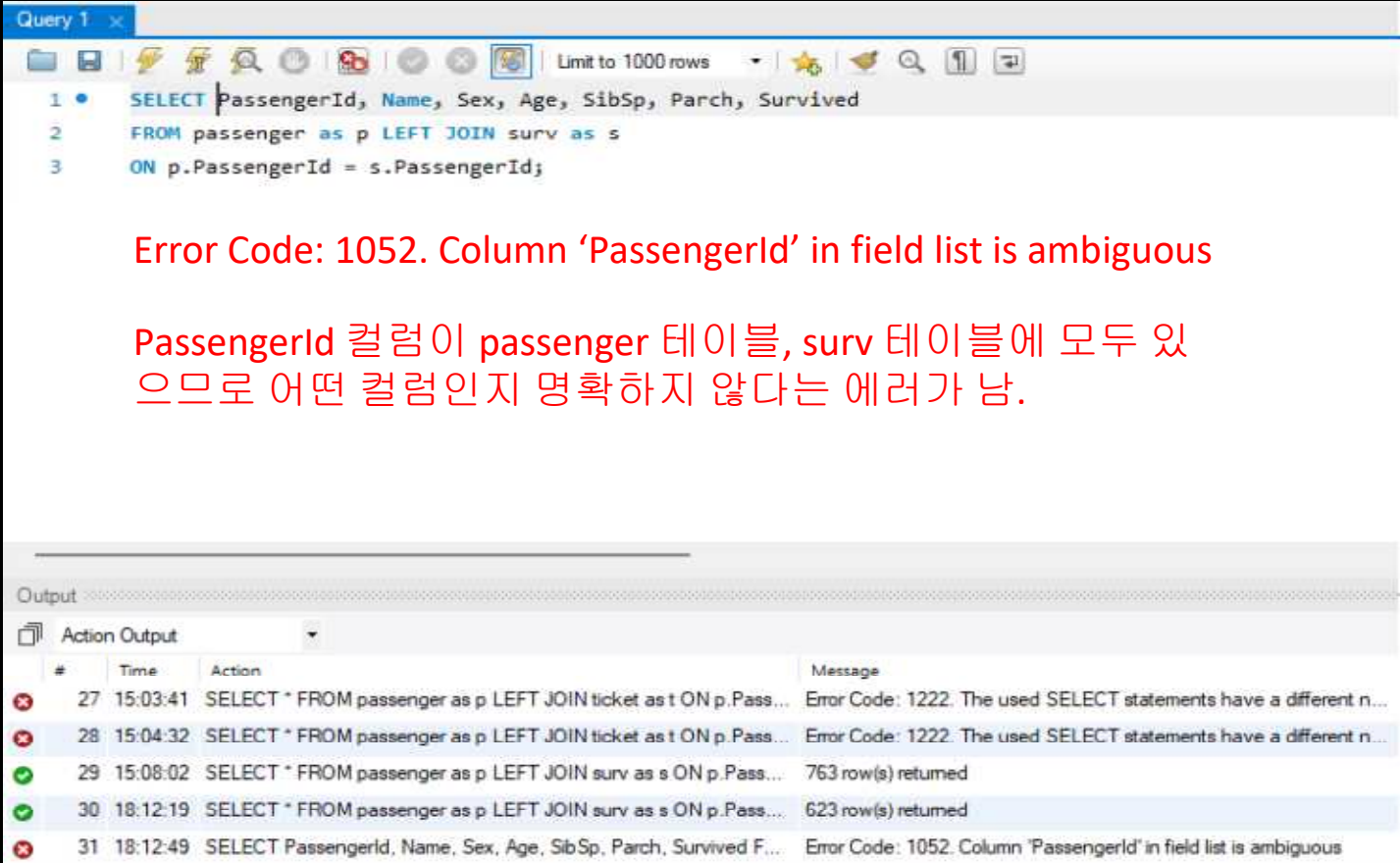
두 개의 테이블을 조인 한 후 원하는 컬럼의 자료를 조회하기 위해서는 **SELECT 테이블명1.컬럼명1, 테이블명2.컬럼명2** 의 형태로 컬럼명이 있는 테이블명을 앞에 명시해 주어야 함.

기준 컬럼(e.g.PassengerId)의 경우 두 테이블 모두에 있으므로 반드시 어느 테이블의 컬럼을 사용할 것인지 지정해주어야 함.

조인 후 원하는 컬럼 검색하기

```
SELECT PassengerId, Name, Sex, Age, SibSp, Parch, Survived FROM passenger AS p LEFT JOIN surv AS s  
ON p.PassengerId = s.PassengerId;
```

passenger 테이블과
surv 테이블을 조인해
서 PassengerId, Name,
Sex, Age, SibSp, Parch,
Survived 컬럼을 조회하
는 쿼리



The screenshot shows a SQL query editor window titled 'Query 1'. The query is:

```
1 • SELECT PassengerId, Name, Sex, Age, SibSp, Parch, Survived
2 FROM passenger as p LEFT JOIN surv as s
3 ON p.PassengerId = s.PassengerId;
```

Below the query, there is a red error message:

Error Code: 1052. Column 'PassengerId' in field list is ambiguous

Below the error message, there is a red text explanation:

PassengerId 컬럼이 passenger 테이블, surv 테이블에 모두 있으므로 어떤 컬럼인지 명확하지 않다는 에러가 남.

At the bottom, there is an 'Output' section with a table showing the execution history:

#	Time	Action	Message
27	15:03:41	SELECT * FROM passenger as p LEFT JOIN ticket as t ON p.Pass...	Error Code: 1222. The used SELECT statements have a different n...
28	15:04:32	SELECT * FROM passenger as p LEFT JOIN ticket as t ON p.Pass...	Error Code: 1222. The used SELECT statements have a different n...
29	15:08:02	SELECT * FROM passenger as p LEFT JOIN surv as s ON p.Pass...	763 row(s) returned
30	18:12:19	SELECT * FROM passenger as p LEFT JOIN surv as s ON p.Pass...	623 row(s) returned
31	18:12:49	SELECT PassengerId, Name, Sex, Age, SibSp, Parch, Survived F...	Error Code: 1052. Column 'PassengerId' in field list is ambiguous

조인 후 원하는 컬럼 검색하기

```
SELECT p.PassengerId, Name, Sex, Age, SibSp, Parch, Survived FROM passenger AS p LEFT JOIN surv AS s  
ON p.PassengerId = s.PassengerId;
```

passenger 테이블과 surv 테이블을 조인
해서 PassengerId, Name, Sex, Age, SibSp,
Parch, Survived 컬럼을 조회하는 쿼리

PassengerId가 양쪽 테이블 모두에 있기
때문에 passenger 테이블의 PassengerId
를 가져오도록 p.PassengerId로 컬럼명
을 명시

Query 1

```
1 • SELECT p.PassengerId, Name, Sex, Age, SibSp, Parch, Survived  
2 FROM passenger as p LEFT JOIN surv as s  
3 ON p.PassengerId = s.PassengerId;
```

passenger 테이블의 PassengerId 컬럼으로 지정해서 조회

Result Grid

	PassengerId	Name	Sex	Age	SibSp	Parch	Survived
▶	193	Andersen-Jensen, Miss. Carla Christine Nielsine	female	19	1	0	1
	192	Carbines, Mr. William	male	19	0	0	HULL
	715	Greenberg, Mr. Samuel	male	52	0	0	HULL
	533	Elias, Mr. Joseph Jr	male	17	1	1	HULL
	133	Robins, Mrs. Alexander A (Grace Charity Laury)	female	47	1	0	HULL
	400	Trout, Mrs. William H (Jessie L)	female	28	0	0	1

Result 18

Output

Action Output

#	Time	Action	Message
✖	31 18:12:49	SELECT PassengerId, Name, Sex, Age, SibSp, Parch, Survived F...	Error Code: 1052. Column
✔	32 18:20:12	SELECT p.PassengerId, Name, Sex, Age, SibSp, Parch, Survived ...	623 row(s) returned
✔	33 18:21:03	SELECT p.PassengerId, p.Name, p.Sex, p.Age, p.SibSp, p.Parch, ...	623 row(s) returned
✔	34 18:22:02	SELECT p.PassengerId, p.Name, p.Sex, p.Age, p.SibSp, p.Parch, ...	623 row(s) returned
✔	35 18:22:16	SELECT p.PassengerId, Name, Sex, Age, SibSp, Parch, Survived ...	623 row(s) returned

3개 이상의 테이블 조인하기

passenger, ticket, survived 테이블을 INNER JOIN으로 묶어 전체 컬럼을 출력하시오.

```
SELECT * FROM passenger AS p  
INNER JOIN ticket AS t ON p.PassengerId = t.PassengerId  
INNER JOIN survived AS s ON p.PassengerId = s.PassengerId;
```

3개 이상의 테이블 조인하기

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows a tree view of databases, with 'titanic' selected. Under 'titanic', the 'Tables' list includes 'p_info', 'passenger', 'surv', 'survived', 't_info', and 'ticket'. The 'passenger' table is highlighted.

The 'Query 1' editor shows the following SQL query:

```
1 SELECT * FROM passenger AS p
2 INNER JOIN ticket AS t ON p.PassengerId = t.PassengerId
3 INNER JOIN survived AS s ON p.PassengerId = s.PassengerId;
```

The 'Result Grid' shows the results of the query. The columns are: PassengerId, Name, Sex, Age, SibSp, Parch, PassengerId, Ticket, Pclass, Fare, and C. The results are as follows:

	PassengerId	Name	Sex	Age	SibSp	Parch	PassengerId	Ticket	Pclass	Fare	C
▶	1	Braund, Mr. Owen Harris	male	22	1	0	1	A/5 21171	3	7.25	NU
	3	Heikkinen, Miss. Laina	female	26	0	0	3	STON/O2. 3101282	3	7.925	NU
	4	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	4	113803	1	53.1	C
	5	Allen, Mr. William Henry	male	35	0	0	5	373450	3	8.05	NU
	9	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27	0	2	9	347742	3	11.1333	NU

The 'Output' pane shows the 'Action Output' for the query. The results are as follows:

#	Time	Action	Message	Duration / Fetch
✓ 41	18:42:13	SELECT * FROM ticket AS t INNER JOIN surv AS s ON t.Passeng...	0 row(s) returned	0.016 sec / 0.000 sec
✓ 42	18:43:44	SELECT * FROM passenger AS p INNER JOIN ticket AS t ON p.P...	317 row(s) returned	0.016 sec / 0.000 sec
✓ 43	18:51:09	EXPLAIN SELECT * FROM passenger AS p INNER JOIN ticket A...	OK	0.000 sec
✓ 44	18:51:09	EXPLAIN FORMAT=JSON SELECT * FROM passenger AS p INN...	OK	0.000 sec
✓ 45	18:52:00	SELECT * FROM passenger AS p INNER JOIN ticket AS t ON p.P...	317 row(s) returned	0.015 sec / 0.000 sec

조인 후 원하는 조건의 컬럼 검색하기

```
SELECT p.PassengerId, Name, Sex, Age, SibSp, Parch, Survived FROM passenger AS  
p LEFT JOIN surv AS s ON p.PassengerId = s.PassengerId;
```

passenger 테이블과 surv 테이블을 조인
해서 PassengerId, Name, Sex, Age, SibSp,
Parch, Survived 컬럼을 조회하는 쿼리

PassengerId가 양쪽 테이블 모두에 있기
때문에 passenger 테이블의 PassengerId
를 가져오도록 p.PassengerId로 컬럼명
을 명시

Query 1

```
1 • SELECT p.PassengerId, Name, Sex, Age, SibSp, Parch, Survived  
2 FROM passenger as p LEFT JOIN surv as s  
3 ON p.PassengerId = s.PassengerId;
```

passenger 테이블의 PassengerId 컬럼으로 지정해서 조회

	PassengerId	Name	Sex	Age	SibSp	Parch	Survived
▶	193	Andersen-Jensen, Miss. Carla Christine Nielsine	female	19	1	0	1
	192	Carbines, Mr. William	male	19	0	0	HULL
	715	Greenberg, Mr. Samuel	male	52	0	0	HULL
	533	Elias, Mr. Joseph Jr	male	17	1	1	HULL
	133	Robins, Mrs. Alexander A (Grace Charity Laury)	female	47	1	0	HULL
	400	Trout, Mrs. William H (Jessie L)	female	28	0	0	1

Result 18

Output

Action Output

#	Time	Action	Message
✖	31 18:12:49	SELECT PassengerId, Name, Sex, Age, SibSp, Parch, Survived F...	Error Code: 1052. Column 'PassengerId' is not valid in the context of the query.
✔	32 18:20:12	SELECT p.PassengerId, Name, Sex, Age, SibSp, Parch, Survived ...	623 row(s) returned
✔	33 18:21:03	SELECT p.PassengerId, p.Name, p.Sex, p.Age, p.SibSp, p.Parch, ...	623 row(s) returned
✔	34 18:22:02	SELECT p.PassengerId, p.Name, p.Sex, p.Age, p.SibSp, p.Parch, ...	623 row(s) returned
✔	35 18:22:16	SELECT p.PassengerId, Name, Sex, Age, SibSp, Parch, Survived ...	623 row(s) returned

데이터 분석을 위한 SQL입문 연습문제 10

1. passenger, ticket, survived 테이블을 조인하고 Survived가 1인 사람들만 찾아서 Name, Age, Sex, Pclass, survived 컬럼을 출력하시오.
2. 1의 결과를 10개만 출력하시오.
3. Passenger 테이블을 기준 ticket, survived테이블을 LEFT JOIN 한 결과에서 성별이 여성이면서 Pclass가 1인 사람 중 생존자(survived=1)를 찾아 이름, 성별, Pclass를 표시하시오.

데이터 분석을 위한 SQL입문 연습문제 10

4. passenger, ticket, survived 테이블을 left join 후 나이가 10세 이상 20세 이하 이면서 Pclass 2인 사람 중 생존자를 표시하시오.
5. passenger, ticket, survived 테이블을 left join 후 성별이 여성 또는 Pclass 가 1인 사람 중 생존자를 표시하시오.
6. passenger, ticket, survived 테이블을 left join 후 생존자 중에서 이름에 Mrs가 포함된 사람을 찾아 이름, Pclass, 나이, Parch, Survived 를 표시하시오.

데이터 분석을 위한 SQL입문 연습문제 10

7. passenger, ticket, survived 테이블을 left join 후 Pclass가 1, 2이고 Embarked가 s, c 인 사람중에서 생존자를 찾아 이름, 성별, 나이를 표시하시오.
8. passenger, ticket, survived 테이블을 left join 후 이름에 James가 들어간 사람중 생존자를 찾아 이름, 성별, 나이를 표시하고 나이를 기준으로 내림차순 정렬하시오.
9. passenger, ticket, survived 테이블을 INNER JOIN한 데이터에서 성별별, 생존자의 숫자를 구하시오. 생존자 숫자 결과는 별칭을 Total로 하시오.

데이터 분석을 위한 SQL입문 연습문제 10

10. passenger, ticket, survived 테이블을 INNER JOIN한 데이터에서 성별별, 생존자의 숫자, 생존자 나이의 평균을 구하시오. 생존자 숫자 결과는 별칭을 Total로 하시오.

11. passenger, ticket, survived 테이블을 INNER JOIN한 데이터에서 성별별, pclass별, 생존자별로 pclass, sex, survived, survived의 클래스별 합계, 생존자/사망자의 나이 평균을 구하시오.

survived의 별칭은 is_survived로, 생존자 클래스별 합계는 별칭을 survived_total로, 생존자/사망자의 나이 평균은 별칭을 avg_age로 하시오.