

SQL 기초 문법

SELECT

SQL 기초 문법 | SQL 코딩 테스트 준비하기

강사 나동빈

SQL 기초 문법

SELECT

SQL 기초 문법 SELECT

예제 테이블 확인하기

SQL.
기초 다지기

- 실습을 위해 테이블을 생성한다.

```
DROP TABLE IF EXISTS course;  
CREATE TABLE course (  
    course_id INT PRIMARY KEY,  
    course_name VARCHAR(50) NOT NULL,  
    course_cost INT NOT NULL,  
    course_date DATE  
);
```

SQL 기초 문법

SELECT

예제 테이블 확인하기

SQL.
기초 다지기

- 실습을 위해 다수의 레코드(record)를 삽입한다.

```
INSERT INTO course values (1, 'Programming', 30000, '2015-09-07');  
INSERT INTO course values (2, 'MySQL', 40000, '2015-09-10');  
INSERT INTO course values (3, 'Oracle 11g', 35000, '2015-09-09');  
INSERT INTO course values (4, 'Physics', 20000, '2015-09-10');  
INSERT INTO course values (5, 'Education', 30000, '2015-09-09');  
INSERT INTO course values (6, 'Economics', 40000, '2015-09-10');
```

SQL 기초 문법 SELECT

예제 테이블 확인하기

SQL.
기초 다지기

- 실습을 위해 다수의 레코드(record)를 삽입한다.

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

```

1. INSERT INTO course values (1, 'Programming', 30000, '2015-09-07');
2. INSERT INTO course values (2, 'MySQL', 40000, '2015-09-10');
3. INSERT INTO course values (3, 'Oracle 11g', 35000, '2015-09-09');
4. INSERT INTO course values (4, 'Physics', 20000, '2015-09-10');
5. INSERT INTO course values (5, 'Education', 30000, '2015-09-09');
6. INSERT INTO course values (6, 'Economics', 40000, '2015-09-10');
7. SELECT * FROM course;
  
```

Below the queries, the result grid displays the data inserted into the 'course' table:

course_id	course_name	course_cost	course_date
1	Programming	30000	2015-09-07
2	MySQL	40000	2015-09-10
3	Oracle 11g	35000	2015-09-09
4	Physics	20000	2015-09-10
5	Education	30000	2015-09-09
6	Economics	40000	2015-09-10

SQL 기초 문법 SELECT

SELECT 명령어

SQL.
기초 다지기

- SELECT 명령어를 이용하여 데이터를 조회할 수 있다.

```
SELECT {컬럼명} FROM {테이블명} {조건문};
```

SQL 기초 문법 SELECT

SELECT 명령어

SQL.
기초 다지기

- course_cost 컬럼의 값이 35,000이거나 40,000인 데이터를 조회할 수 있다.
- 컬럼명에 별표(*) 기호를 넣으면, 모든 컬럼을 조회한다.

```
SELECT * FROM course WHERE course_cost = 40000 OR course_cost = 35000;
```

Query 1 x

Limit to 1000 rows

```

1 • SELECT *
2   FROM course
3   WHERE course_cost = 40000 OR course_cost = 35000;
4

```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content: |

	course_id	course_name	course_cost	course_date
▶	2	MySQL	40000	2015-09-10
	3	Oracle 11g	35000	2015-09-09
	6	Economics	40000	2015-09-10
*	NULL	NULL	NULL	NULL

SQL 기초 문법

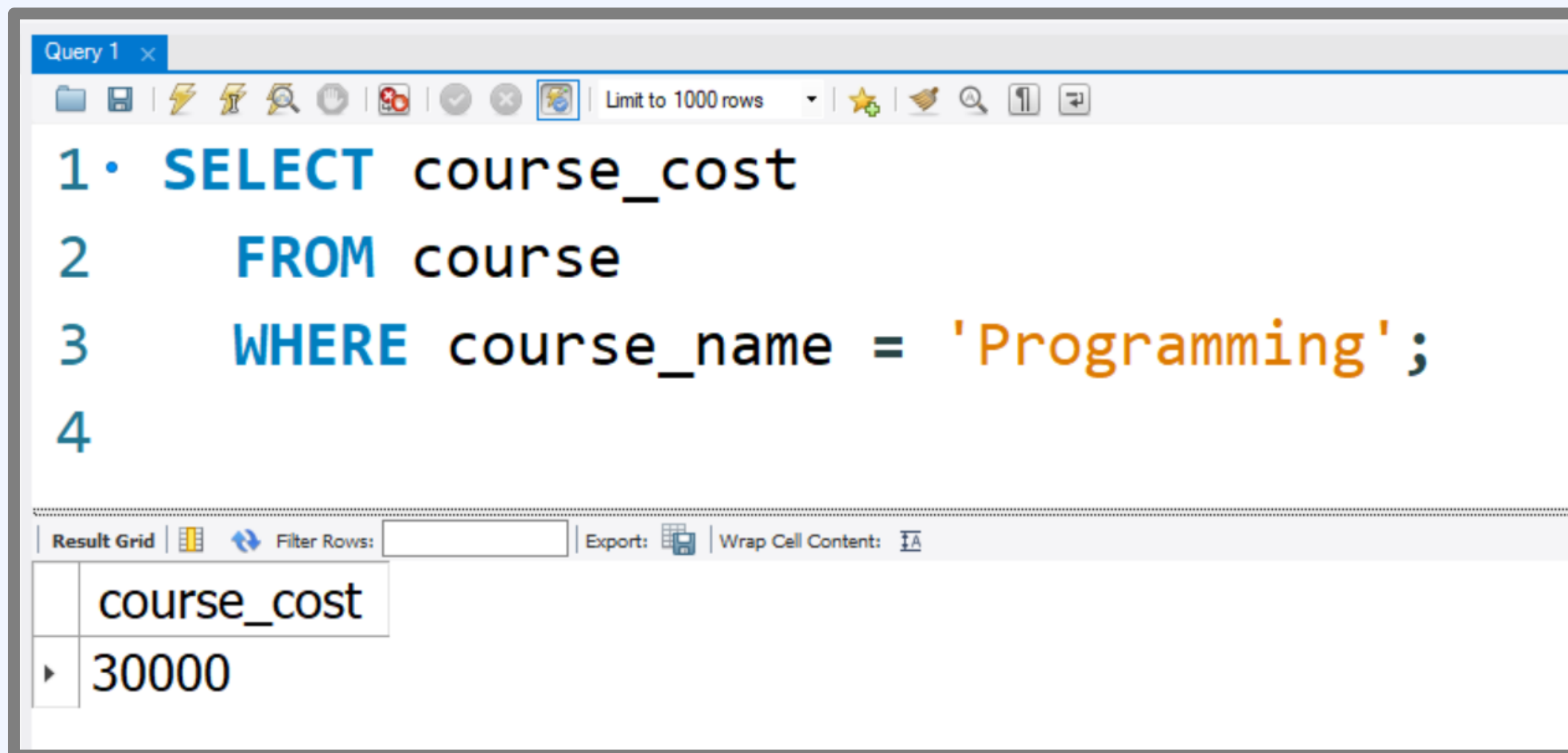
SELECT

SELECT 명령어

SQL.
기초 다지기

- 특정한 과목 이름을 갖는 강의의 비용을 조회할 수 있다.

```
SELECT course_cost FROM course WHERE course_name = 'Programming';
```



SQL 기초 문법

SELECT

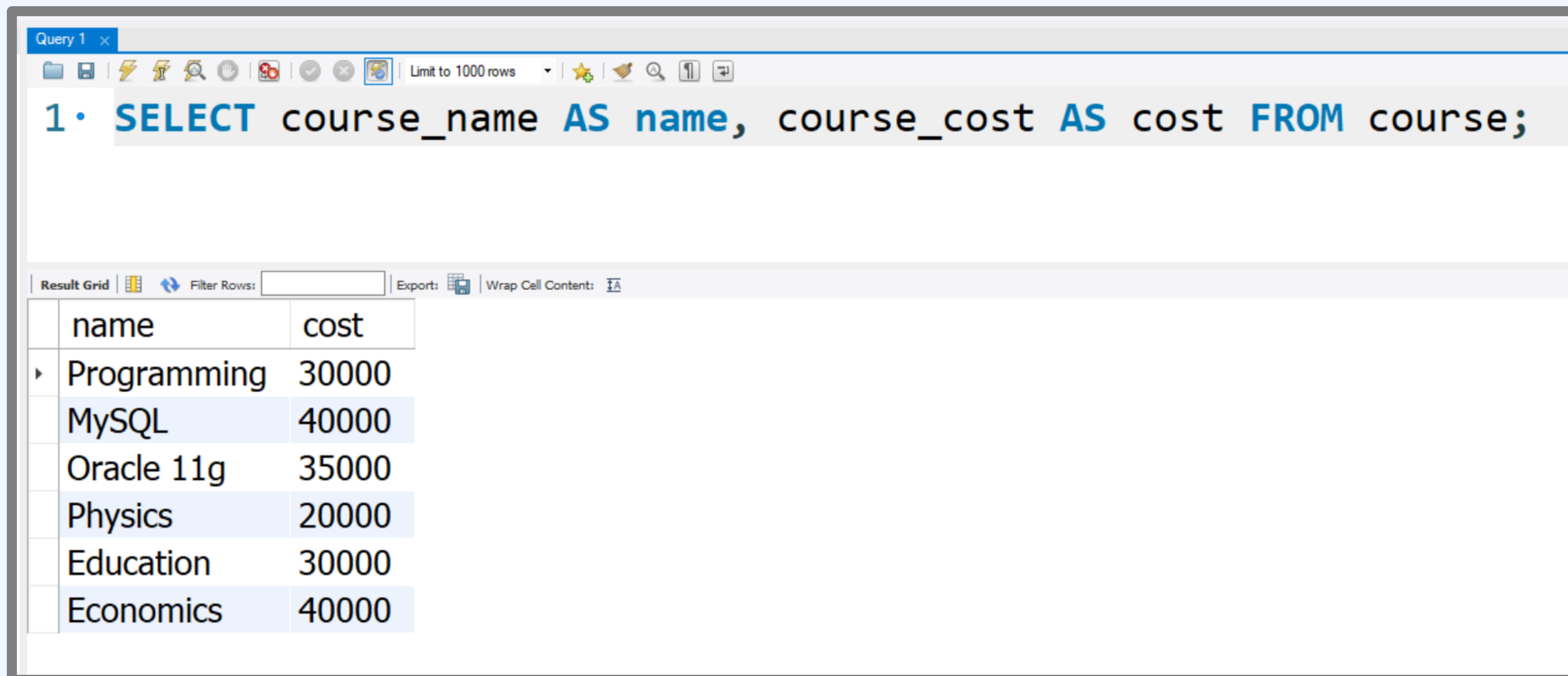
AS를 이용한 별칭(Alias) 부여

SQL.

기초 다지기

- SELECT 구문에 AS를 사용하여 조회를 간편하게 할 수 있다.
- 복잡한 SQL 구문에서 전체 코드 길이를 짧게 만들 수 있다.

```
SELECT course_name AS name, course_cost AS cost FROM course;
```



The screenshot shows a SQL query editor window titled 'Query 1'. The query text is: `1. SELECT course_name AS name, course_cost AS cost FROM course;`. Below the query, there is a 'Result Grid' section. It includes a 'Filter Rows' input field and an 'Export' button. The results are displayed in a table with two columns: 'name' and 'cost'. The table contains six rows of data.

name	cost
Programming	30000
MySQL	40000
Oracle 11g	35000
Physics	20000
Education	30000
Economics	40000

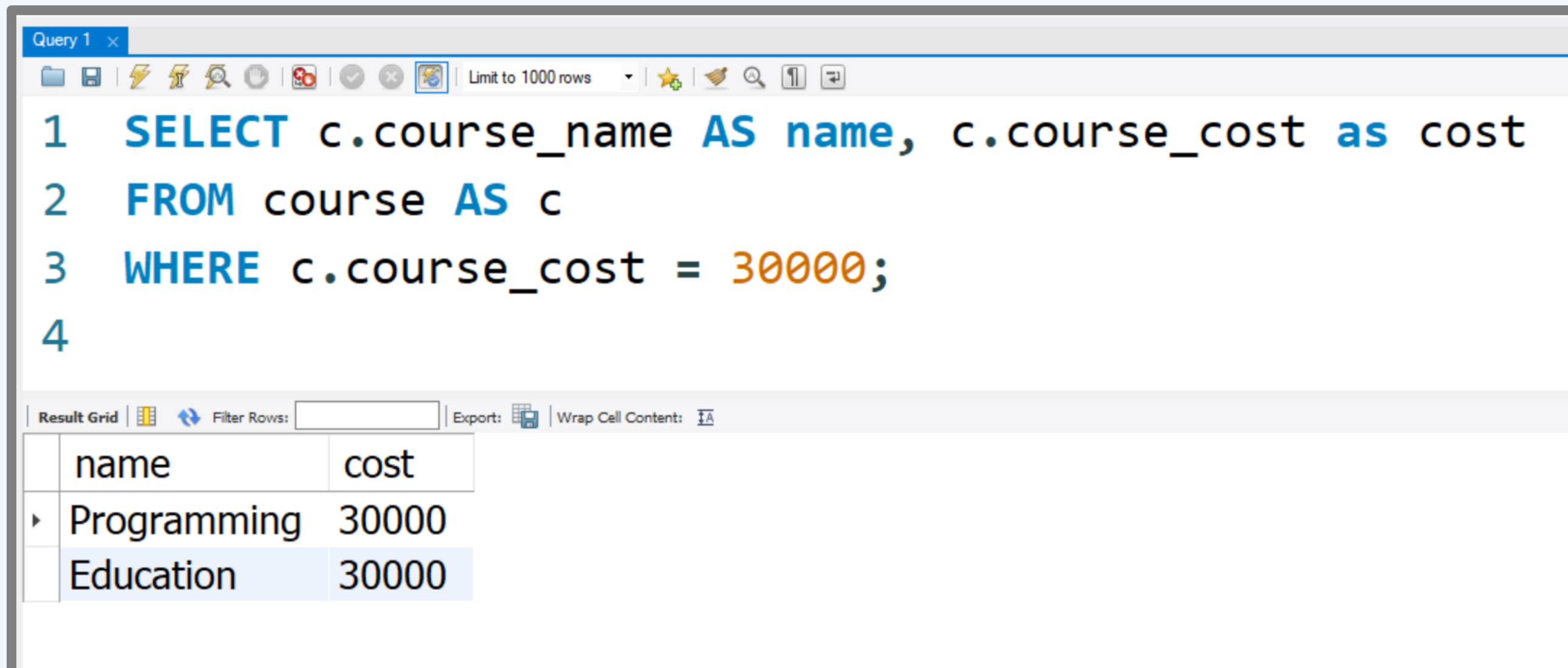
SQL 기초 문법 SELECT

AS를 이용한 별칭(Alias) 부여

SQL.
기초 다지기

- 테이블에 대해서도 별칭(alias)을 사용할 수 있다.

```
SELECT c.course_name AS name, c.course_cost AS cost  
FROM course AS c  
WHERE c.course_cost = 30000;
```



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

```
1 SELECT c.course_name AS name, c.course_cost as cost  
2 FROM course AS c  
3 WHERE c.course_cost = 30000;  
4
```

Below the query, there is a 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The results are displayed in a table with two columns: 'name' and 'cost'.

name	cost
▶ Programming	30000
Education	30000

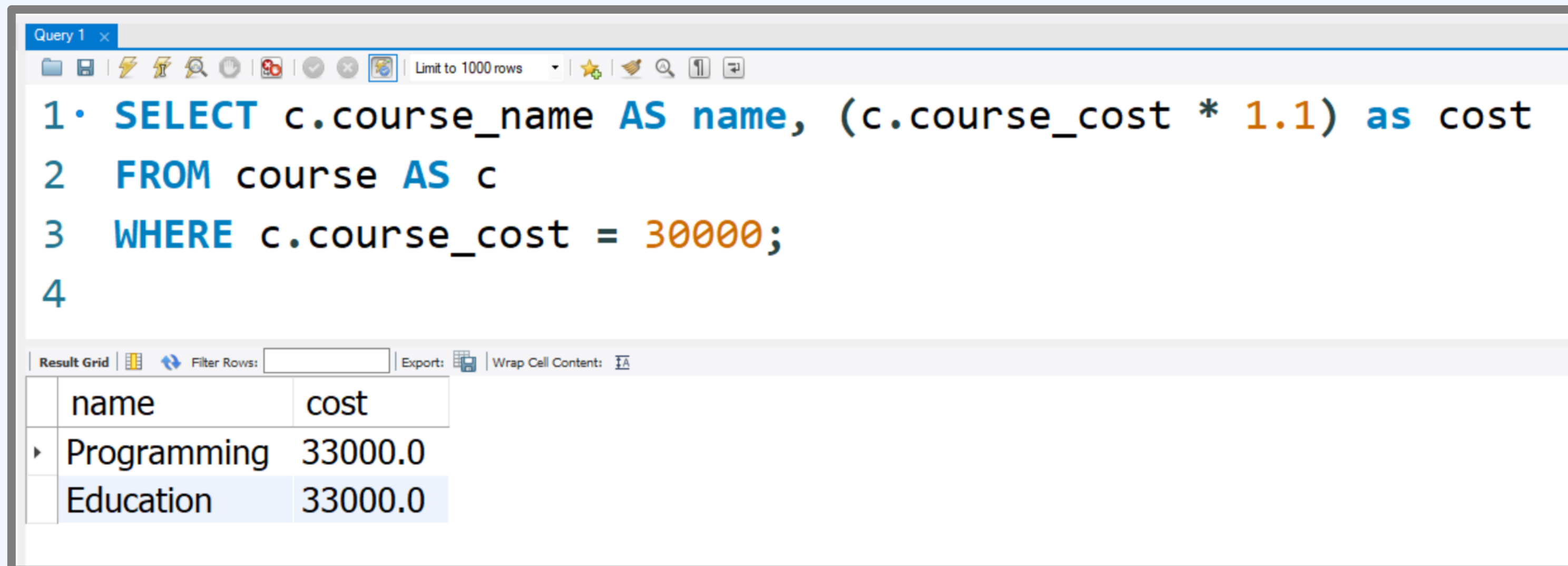
SQL 기초 문법 SELECT

AS를 이용한 별칭(Alias) 부여

SQL. 기초 다지기

- 테이블에 대해서도 별칭(alias)을 사용할 수 있다.

```
SELECT c.course_name AS name, (c.course_cost * 1.1) AS cost
FROM course AS c
WHERE c.course_cost = 30000;
```



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

```
1. SELECT c.course_name AS name, (c.course_cost * 1.1) as cost
2 FROM course AS c
3 WHERE c.course_cost = 30000;
4
```

Below the query, the 'Result Grid' is displayed, showing the results of the query. The grid has two columns: 'name' and 'cost'. The results are:

name	cost
Programming	33000.0
Education	33000.0

SQL 기초 문법 SELECT

ORDER BY를 활용한 정렬

SQL.
기초 다지기

- 조회된 데이터를 특정한 기준으로 정렬할 수 있다.

```
SELECT {컬럼명} FROM {테이블명} ORDER BY {기준 컬럼명} {DESC 혹은 ASC};
```

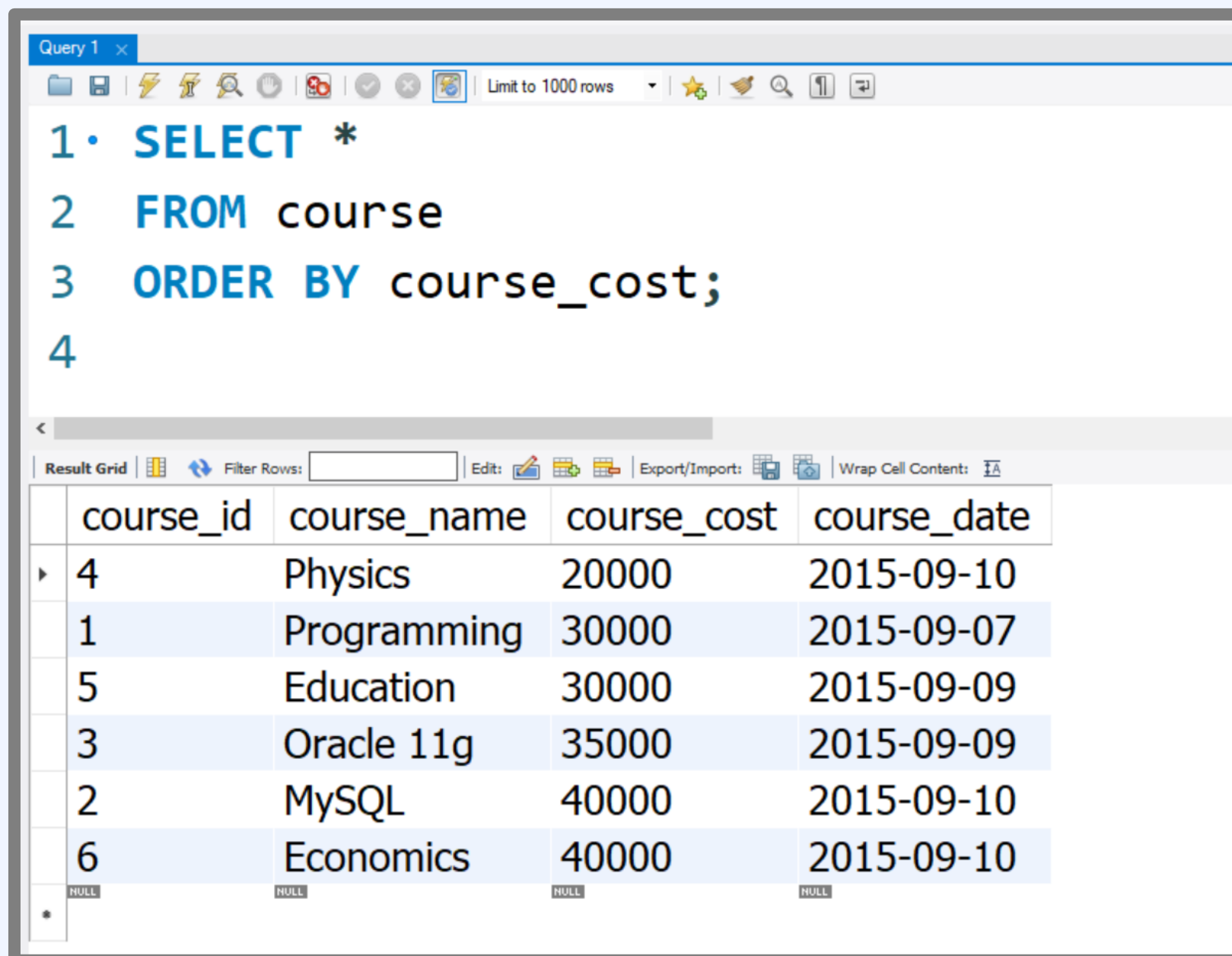
SQL 기초 문법 SELECT

ORDER BY를 활용한 정렬

SQL.
기초 다지기

- 각 강의를 강의 비용 순서대로 정렬할 수 있다.

```
SELECT *
FROM course
ORDER BY course_cost;
```



	course_id	course_name	course_cost	course_date
▶	4	Physics	20000	2015-09-10
	1	Programming	30000	2015-09-07
	5	Education	30000	2015-09-09
	3	Oracle 11g	35000	2015-09-09
	2	MySQL	40000	2015-09-10
	6	Economics	40000	2015-09-10

SQL 기초 문법 SELECT

ORDER BY를 활용한 정렬

SQL.
기초 다지기

- 조회된 데이터를 정렬할 때 DESC 옵션으로 내림차순(descending) 정렬할 수 있다.

```
SELECT * FROM course ORDER BY course_cost DESC;
```

Query 1

Limit to 1000 rows

```
1. SELECT *
2. FROM course
3. ORDER BY course_cost DESC;
```

Result Grid

	course_id	course_name	course_cost	course_date
▶	2	MySQL	40000	2015-09-10
	6	Economics	40000	2015-09-10
	3	Oracle 11g	35000	2015-09-09
	1	Programming	30000	2015-09-07
	5	Education	30000	2015-09-09
	4	Physics	20000	2015-09-10
*	NULL	NULL	NULL	NULL

SQL 기초 문법 SELECT

ORDER BY를 활용한 정렬

SQL.
기초 다지기

- 기준이 여러 개일 때도 정렬할 수행할 수 있다.

```
SELECT * FROM course ORDER BY course_cost DESC, course_name ASC;
```

Query 1

Limit to 1000 rows

```
1. SELECT *
2. FROM course
3. ORDER BY course_cost DESC, course_name ASC;
4.
```

Result Grid

	course_id	course_name	course_cost	course_date
▶	6	Economics	40000	2015-09-10
	2	MySQL	40000	2015-09-10
	3	Oracle 11g	35000	2015-09-09
	5	Education	30000	2015-09-09
	1	Programming	30000	2015-09-07
	4	Physics	20000	2015-09-10
	NULL	NULL	NULL	NULL

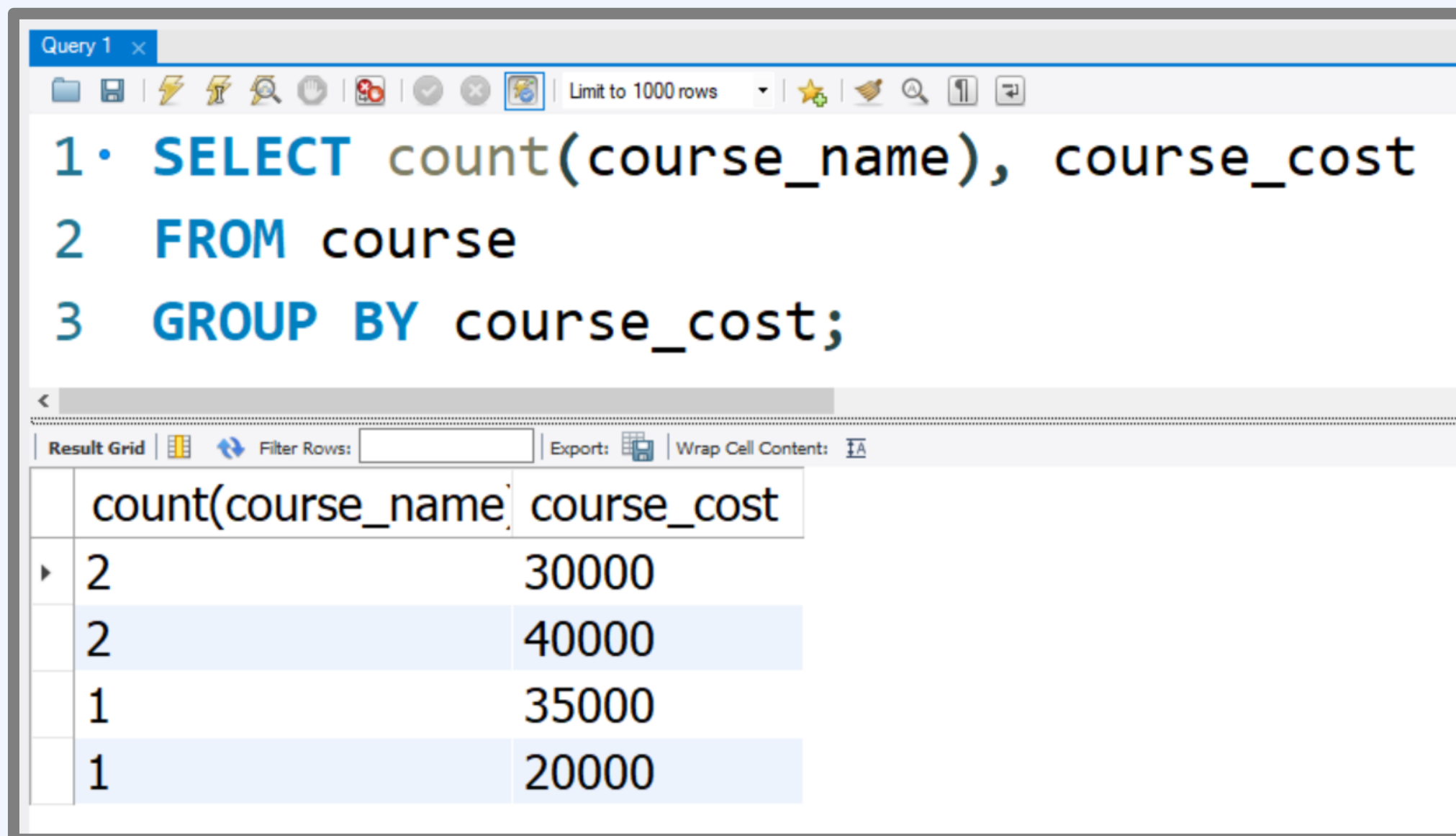
SQL 기초 문법 SELECT

GROUP BY 명령어

SQL.
기초 다지기

- 수강 비용이 같은 강의들에 대하여 그 개수를 구할 수 있다.
- 집계 함수로는 COUNT(), SUM() 등이 있다.

```
SELECT count(course_name), course_cost FROM course GROUP BY course_cost;
```



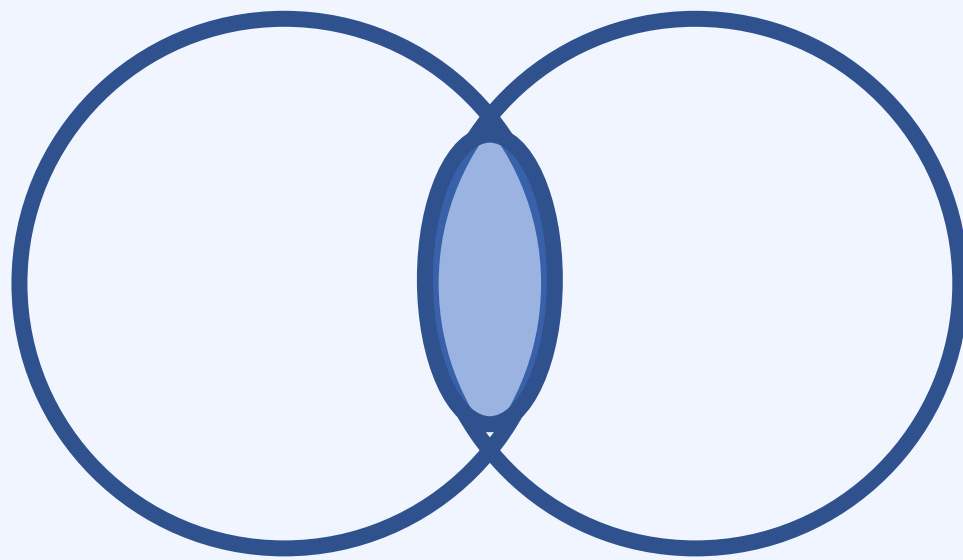
	count(course_name)	course_cost
2	2	30000
2	2	40000
1	1	35000
1	1	20000

SQL 기초 문법 SELECT

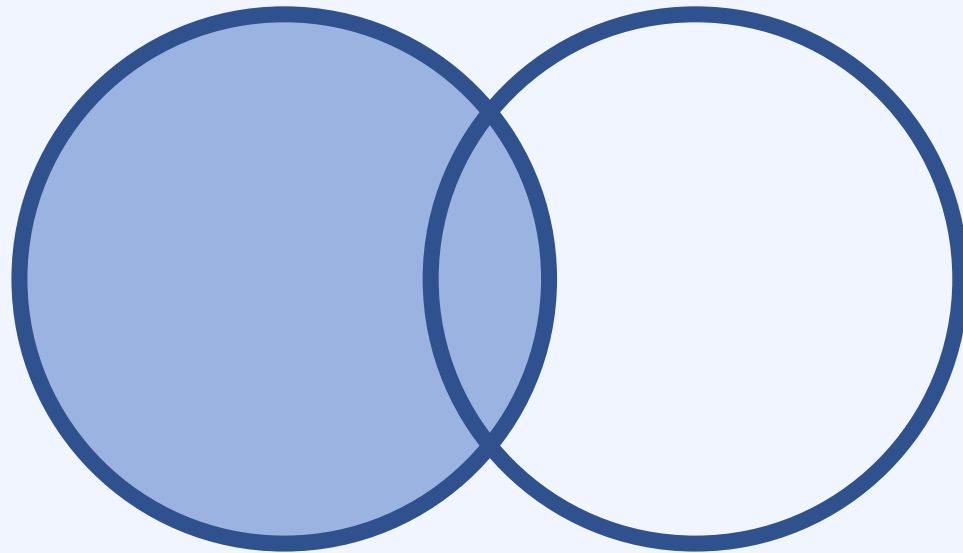
JOIN 명령어

SQL. 기초 다지기

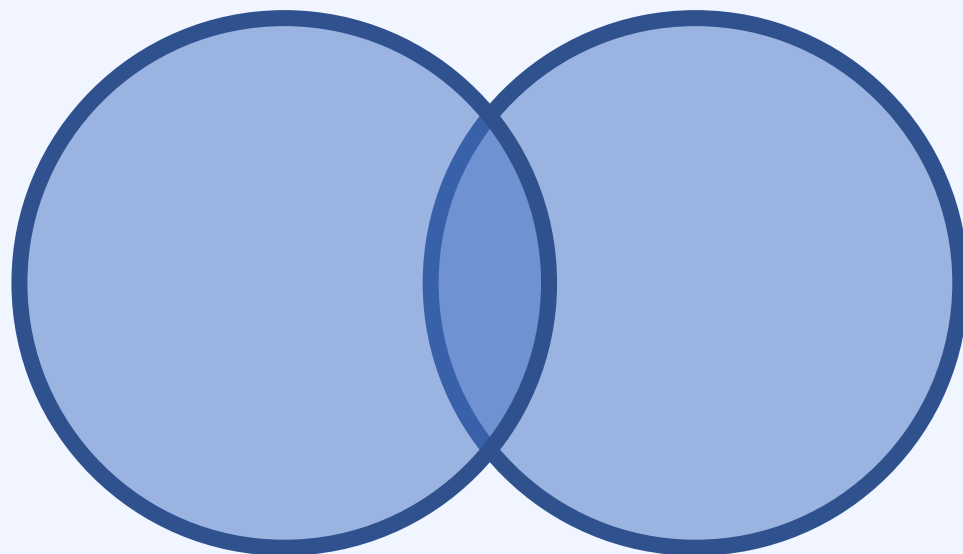
- JOIN 명령어를 이용하여 두 테이블의 레코드를 매칭한 결과를 얻을 수 있다.



INNER JOIN: 테이블 A와 테이블 B에 동일한 Key가 있는 레코드에 대해서만 조회



LEFT JOIN: 테이블 A의 모든 레코드를 조회하되, 테이블 B에 동일한 Key가 있는 레코드와 매칭



FULL OUTER JOIN: 테이블 A와 B의 모든 레코드를 조회하되, 동일한 Key가 있는 레코드는 매칭

SQL 기초 문법 SELECT

JOIN 명령어

SQL.
기초 다지기

- 실습을 위하여 두 개의 테이블을 고려한다.

학생(student)

학생 번호	학생 이름	학생 나이
1	홍길동	20
2	김철수	22
3	이순신	25
4	나동빈	30
5	장영실	29
6	장국영	22

소속(belonging)

학생 번호	학과 이름
1	컴퓨터공학과
2	전자공학과
3	식품영양학과
4	간호학과
4	기계공학과
7	산업디자인과

SQL 기초 문법

SELECT

JOIN 명령어

SQL.

기초 다지기

- 실습을 위하여 두 개의 테이블을 고려한다.

```
CREATE TABLE student (  
    student_id INT PRIMARY KEY,  
    name VARCHAR(20) NOT NULL,  
    age INT NOT NULL  
);
```

```
INSERT INTO student VALUE (1, '홍길동', 20);  
INSERT INTO student VALUE (2, '김철수', 22);  
INSERT INTO student VALUE (3, '이순신', 25);  
INSERT INTO student VALUE (4, '나동빈', 30);  
INSERT INTO student VALUE (5, '장영실', 29);  
INSERT INTO student VALUE (6, '장국영', 22);
```

SQL 기초 문법

SELECT

JOIN 명령어

SQL.

기초 다지기

- 실습을 위하여 두 개의 테이블을 고려한다.

```
CREATE TABLE belonging (  
    student_id INT NOT NULL,  
    department_name VARCHAR(20) NOT NULL  
);
```

```
INSERT INTO belonging VALUE (1, '컴퓨터공학과');  
INSERT INTO belonging VALUE (2, '전자공학과');  
INSERT INTO belonging VALUE (3, '식품영양학과');  
INSERT INTO belonging VALUE (4, '간호학과');  
INSERT INTO belonging VALUE (4, '기계공학과');  
INSERT INTO belonging VALUE (7, '산업디자인과');
```

SQL 기초 문법 SELECT

교차 조인(CROSS JOIN)

SQL.
기초 다지기

- 두 테이블에 대하여 카테시안 곱을 수행한 결과를 반환한다.
- 별다른 조건없이 두 테이블의 행을 모두 조합한 결과를 반환한다.
- 반환된 레코드의 수는 (테이블 A의 레코드 수 × 테이블 B의 레코드 수)이다.

```
SELECT *  
FROM student, belonging;
```

SQL 기초 문법

SELECT

내부 조인(INNER JOIN)

SQL.

기초 다지기

- 테이블 A와 테이블 B에 동일한 Key가 있는 레코드에 대해서만 조회한다.

1) 암묵적 표현(implicit notation)

```
SELECT *  
FROM student, belonging  
WHERE student.student_id = belonging.student_id;
```

2) 명시적 표현(explicit notation)

```
SELECT *  
FROM student  
INNER JOIN belonging  
ON student.student_id = belonging.student_id;
```

SQL 기초 문법

SELECT

왼쪽 조인(LEFT JOIN)

SQL.

기초 다지기

- 테이블 A의 모든 레코드를 조회하되, 테이블 B에서 동일한 Key의 레코드와 매칭한다.
- 명시적 표현(explicit notation)을 이용해 다음과 같이 작성할 수 있다.

```
SELECT *  
FROM student  
LEFT JOIN belonging  
ON student.student_id = belonging.student_id;
```

SQL 기초 문법 SELECT

완전 외부 조인(FULL OUTER JOIN)

SQL.
기초 다지기

- 테이블 A와 B의 모든 레코드를 조회하되, 동일한 Key가 있는 레코드는 매칭한다.
- MySQL에서는 완전 외부 조인을 위한 명시적인 SQL 문법을 지원하지 않는다.
- 따라서 UNION 명령어를 사용하여 완전 외부 조인을 수행할 수 있다.

```
SELECT *  
FROM student  
LEFT JOIN belonging  
    ON student.student_id = belonging.student_id  
UNION  
SELECT *  
FROM student  
RIGHT JOIN belonging  
    ON student.student_id = belonging.student_id;
```