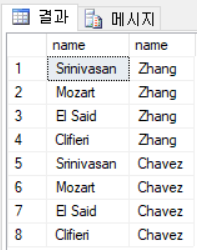
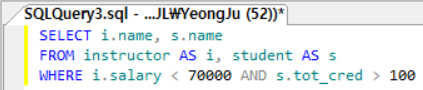
4주차 데이터베이스 실습과제

20175105 곽영주

과제 1 – 1.

1. **재명명 연산**

결과:



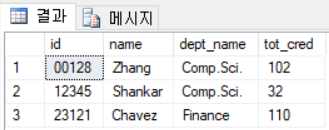
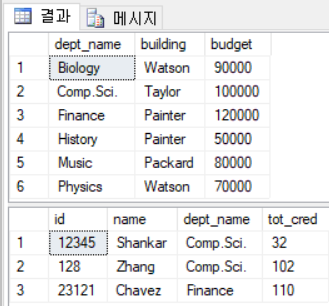
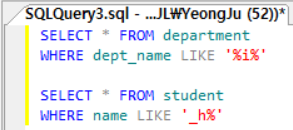
설명:

instructor와 student의 테이블 명을 i와 s로 재명명하여 테이블 명을 간단히 사용하게 만듭니다.

그리고 AND 연산으로 통해 instructor의 salary < 70000와 student의 tot\_cred > 100이란 두 조건이 다 만족하는 데이터 결과를 보여줍니다.

1. **스트링 연산**

결과:



설명:

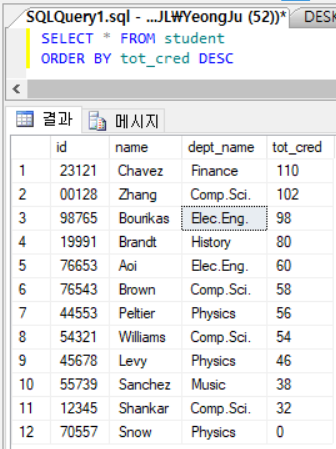
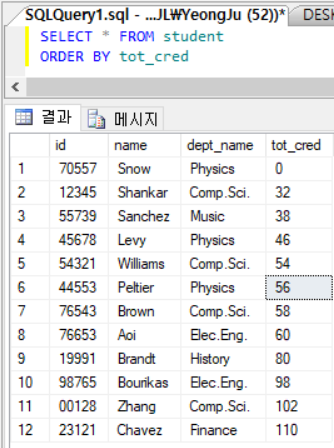
%는 모든 문자를 의미하고 \_는 하나의 문자만 의미합니다.

따라서 ‘%i%’는 앞뒤에 개수 상관없이 모든 문자가 올 수 있고 위치 상관없이 i를 포함하고 있는 데이터 결과를 보여줍니다.

그리고 ‘\_h%’는 뒤는 상관없는데 앞에는 아무 문자 하나만 들어오고 그 다음 h를 포함하고 있는 데이터 결과를 보여줍니다.

1. **데이터 출력의 순서화**

결과:



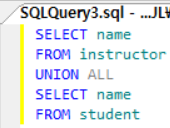
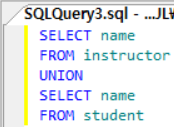
설명:

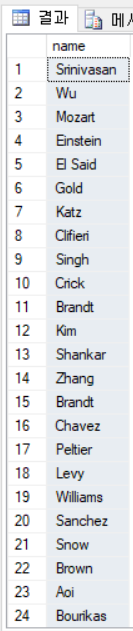
ORDER BY는 데이터 출력의 순서화를 하는 것이고 기본값은 첫번째 결과처럼 오름차순(asc)입니다.

ORDER BY의 기본값은 오름차순(asc)이기 때문에 내림차순(desc)는 두번째 결과처럼 DESC라는 키워드를 적어야 합니다.

1. **UNION, UNION ALL (합집합)**

결과:





설명:

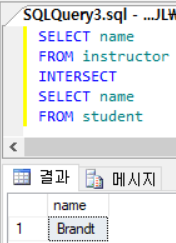
UNION은 중복을 제거한 데이터 결과를 보여줍니다.

UNION ALL은 중복을 제거한 데이터 결과를 보여줍니다.

따라서 위 결과처럼 UNION ALL은 11, 15번 Brandt가 중복되는 것을 볼 수 있습니다.

1. **INTERSECT (교집합)**

결과:



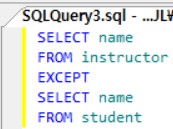
설명:

INTERSECT는 두 테이블의 교집합 데이터 결과를 보여줍니다.

(4)에서 UNION ALL의 결과 중에 Brandt라는 데이터 중복 결과가 있었기 때문에 두 테이블의 INTERSECT 데이터 결과가 Brandt라는 결과를 보여줍니다.

1. **EXCEPT (차집합)**

결과:



설명:

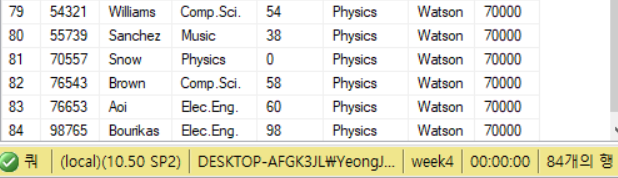
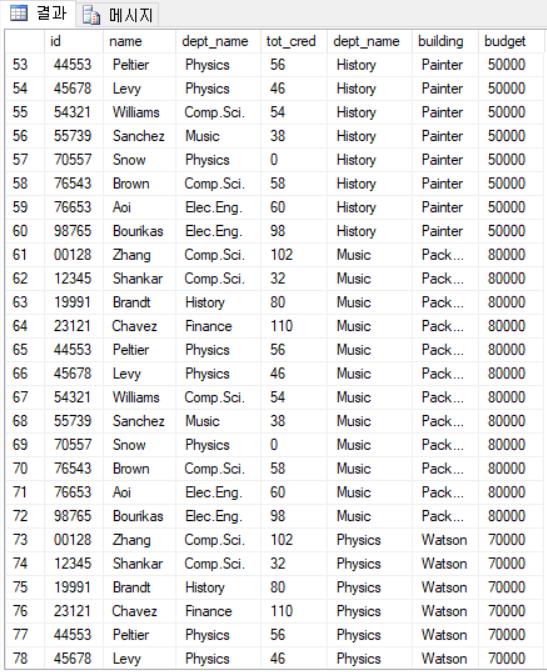
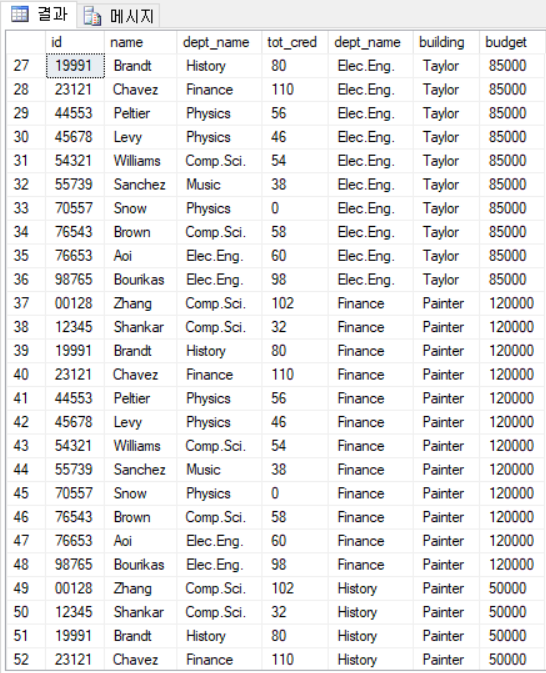
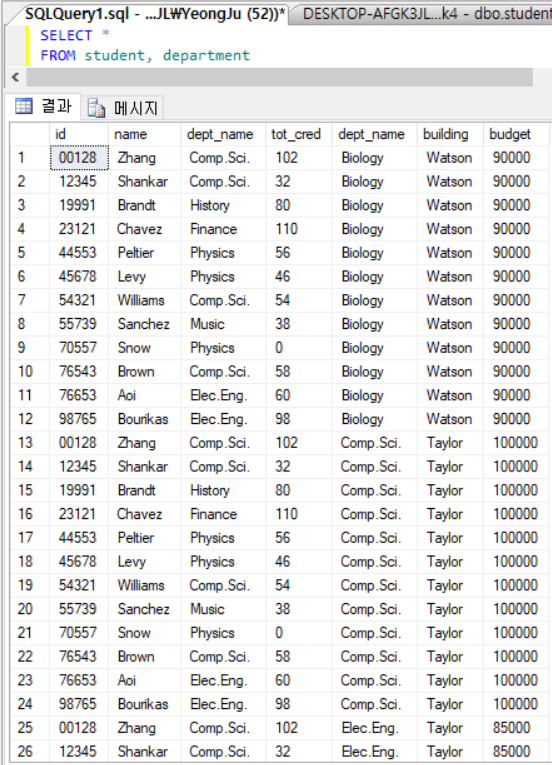
EXCEPT는 차집합을 뜻합니다.

위에서 instructor 데이터에서 에서 student 데이터를 제외한 결과를 보여줍니다.

따라서 겹치던 Brandt 데이터를 제외한 instructor 데이터 결과를 보여줍니다.

1. **카티전 곱**

결과:



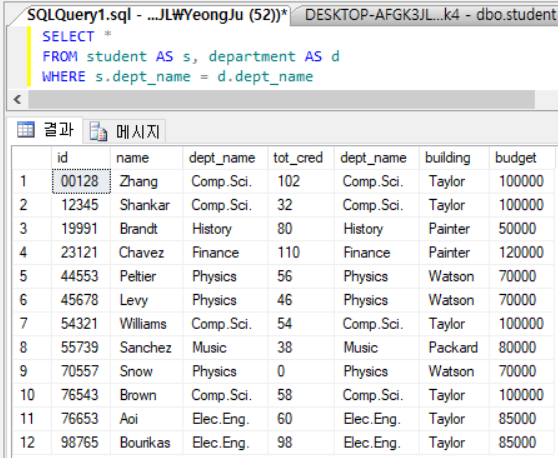
설명:

카티전 곱은 두 테이블 대해서 가능한 모두의 경우의 수를 계산해서 보여줍니다.

따라서 위에 결과처럼 student(12개의 행)과 department(7개의 행)의 모든 경우의 수는 84(12 \* 7)개의 행입니다.

1. **INNER JOIN (WHERE절을 이용)**

결과:

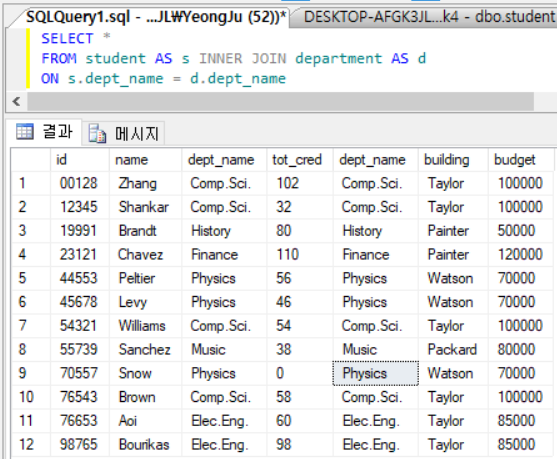


설명:

student와 department 테이블 명을 재명명하고 WHERE절을 이용하여 두 테이블의 dept\_name이라는 공통 컬럼의 값이 같은 데이터만 매치하여 공통 컬럼을 포함하는 데이터 집합의 결과를 보여줍니다.

1. **INNER JOIN (INNER JOIN과 ON을 이용)**

결과:



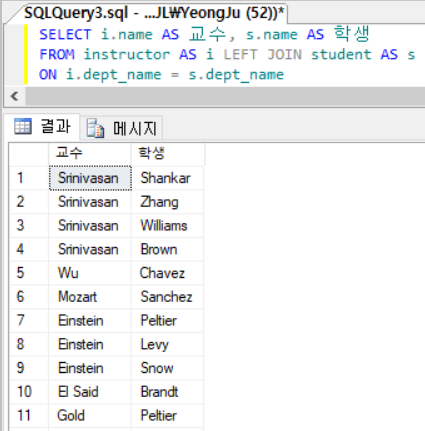
설명:

student와 department 테이블 명을 재명명하고 INNER JOIN과 ON을 이용하여 두 테이블의 dept\_name이라는 공통 컬럼의 값이 같은 데이터만 매치하여 공통 컬럼을 포함하는 데이터 집합의 결과를 보여줍니다.

과제 1 – 2.

1. **OUTER JOIN (LEFT)**

결과:

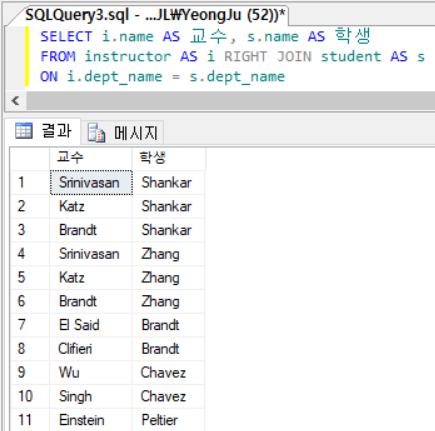
설명:

instructor, student를 i, s로 i.name, s.name을 교수, 학생으로 재명명합니다.

결과를 보시면 20번에 NULL값이 있는 이유는 LEFT JOIN을 통해 왼쪽에 있는 instructor 테이블 기준으로 student에는 같은 dept\_name이 존재하지 않기 때문입니다.

1. **OUTER JOIN (RIGHT)**

결과:



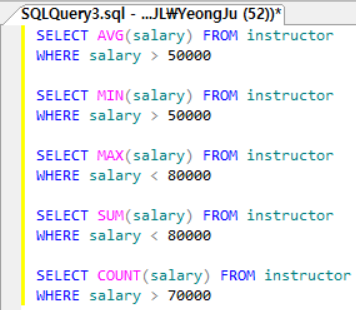
설명:

instructor, student를 i, s로 i.name, s.name을 교수, 학생으로 재명명합니다.

위에 결과에서 (1)에 있던 NULL값이 존재하지 않는 이유는 RIGHT JOIN을 통해 오른쪽에 있는 student 테이블 기준으로 공통 컬럼인 dept\_name의 결과를 보여줬기 때문입니다.

1. **집성함수 (AVG, MIN, MAX, SUM, COUNT)**

결과:

설명:

AVG) instructor의 salary 데이터 중에 50000보다 큰 데이터들의 평균값을 구합니다.

MIN) instructor의 salary 데이터 중에 50000보다 큰 데이터들의 최소값을 구합니다.

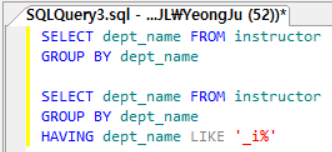
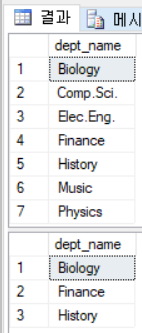
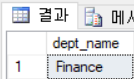
MAX) instructor의 salary 데이터 중에 80000보다 작은 데이터들의 최대값을 구합니다.

SUM) instructor의 salary 데이터 중에 80000보다 작은 데이터들의 합을 구합니다.

COUNT) instructor의 salary 데이터 중에 70000보다 큰 데이터들의 개수를 구합니다.

1. **GROUP BY, HAVING 절**

결과:

설명:

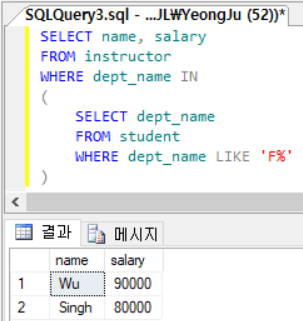
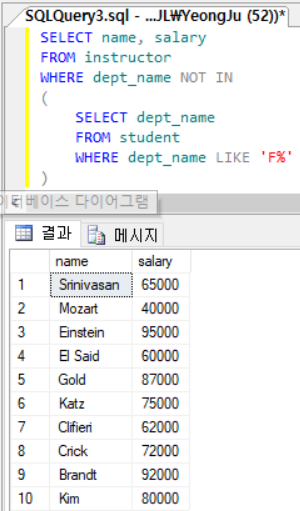
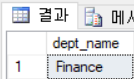
위에 결과에서 GROUP BY를 사용하여 instructor의 dept\_name을 특정 컬럼으로 설정하고 중복 데이터를 제거한 그룹을 데이터를 보여줍니다.

HAVING절은 GROUP BY를 통해 그룹 지어진 데이터에 대한 조건을 부여합니다.

위에 결과처럼 ‘\_i%’라는 조건을 부여하여 i 앞에 하나의 문자만 있는 데이터 그룹을 보여줍니다.

1. **IN & NOT IN**

결과:

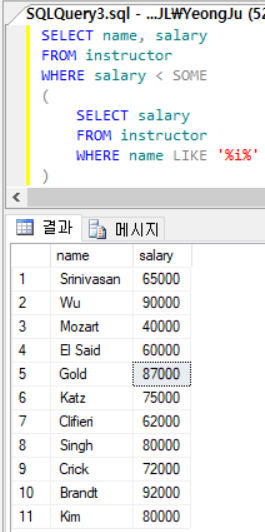
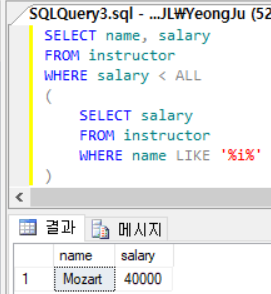
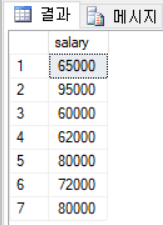
설명:

위에 결과를 보면 IN의 경우는 WHERE의 부질의 부분에서 student의 dept\_name 데이터 중 F로 시작하는 dept\_name 데이터와 같은 instructor의 dept\_name을 포함하는 name, salary결과를 보여줍니다.

하지만 NOT IN의 경우는 WHERE의 부질의 부분에서 student의 dept\_name 데이터 중 F로 시작하는 dept\_name 데이터와 같은 instructor의 dept\_name을 포함하지 않는 name, salary결과를 보여줍니다.

1. **SOME & ALL**

결과:

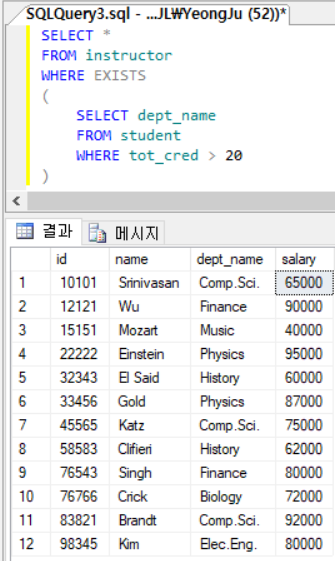
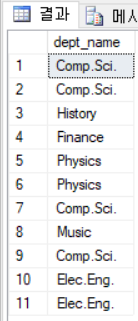
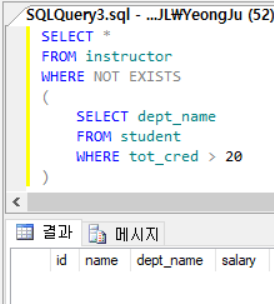
설명:

위에 결과에서 SOME은 WHERE의 부질의 부분에서 instructor의 salary 데이터들 중에 i를 포함하는 데이터를 추출한 데이터들 중에 하나라도 만족하는 instructor의 name, salary의 데이터만 보여줍니다.

ALL은 SOME과 반대로 부질의 부분에서 추출한 데이터들을 모두 만족하는 instructor의 name, salary의 데이터만 보여줍니다.

1. **EXISTS & NOT EXISTS**

결과:

설명:

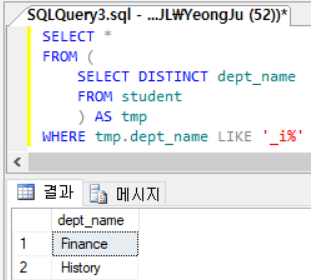
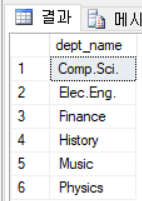
EXISTS는 빈 테이블이 있는지 없는지 검사하는 연산자입니다.

첫번째 결과를 보면 중첩 부질의 부분에 값이 존재하기 때문에 EXISTS는 TRUE이므로 해당되는 instructor 데이터를 보여줍니다.

하지만 NOT EXISTS는 TRUE인 데이터에 NOT을 붙였기 때문에 FALSE이므로 빈 데이터를 보여줍니다.

1. **FROM 절의 부질의**

결과:

설명:

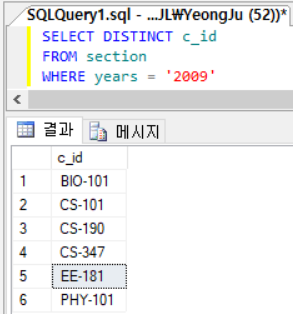
위에 결과에서 중첩 부질의를 FROM에 사용하여 FROM절의 부질의 부분에서 student의 dept\_name 데이터들을 DISTINCT를 통해 중복 제거를 해주고 tmp 재명명을 합니다.

그리고 FROM절의 부질의에서 추출한 데이터에서 i 앞에 하나의 문자만 있는데 데이터들을 보여줍니다.

과제 2.

1. **.**

결과:

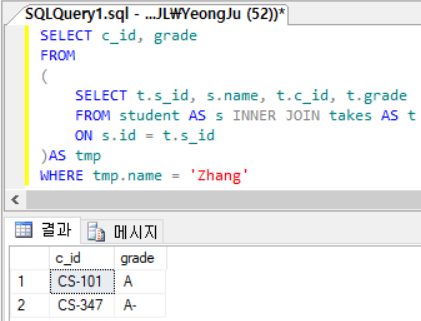


설명:

Section 테이블에서 2009년도 연 모든 과목의 코드를 DISTINCT를 사용하여 중복 없이 검색했습니다.

1. **.**

결과:

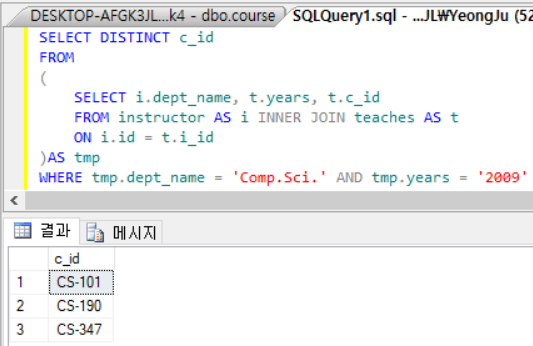


설명:

FROM절의 부질의를 부분에서 student와 takes의 테이블을 학생 코드로 s\_id, name, c\_id, grade 데이터를 INNER JOIN하고 그 데이터 집합에서 ‘Zhang’ 이라는 학생이 들었던 과목의 코드와 성적을 검색했습니다.

1. **.**

결과:

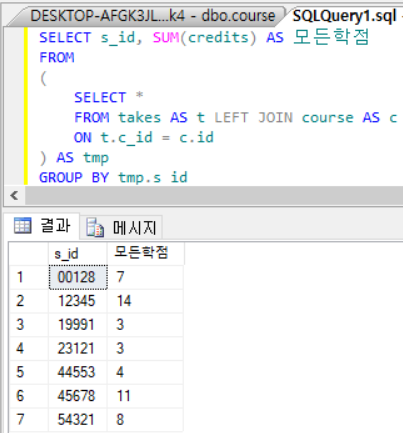


설명:

FROM절의 부질의를 부분에서 instructor와 teaches의 테이블을 강사 코드로 dept\_name, years, c\_id 데이터를 INNER JOIN하고 그 데이터 집합에서 AND 연산을 통해 ‘Comp.Sci.’ 학과에 속한 강사가 진행한 수업 중 2009년에 수업한 과목의 코드를 DISTINCT를 사용하여 중복 제거하고 검색했습니다.

1. **.**

결과:



설명:

FROM절의 부질의 부분에서 takes 테이블 기준으로 course 테이블을 과목 코드로 데이터를 LEFT JOIN하고 GROUP BY를 통해 학생 코드로 그룹 짓고 SUM 연산자를 통해 그룹 지어진 학생 코드에 대한 credits 값들을 더해 총 학점 수(누적된 학점 수)를 구했습니다.