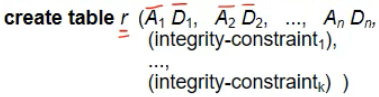
**Chapter 03 – SQL**

**- History**

IBM에서 개발. System R 프로젝트. 채택한 언어가 Sequel 인데 Structured Query Language (SQL)로 이름이 변경됨. 그 뒤 ANSI랑 ISO에서 표준으로 채택됨.

**- *Create Table* Construct**

문법: 

r은 relation 이름, A\_i는 r 스키마의 attribute 이름, D\_i는 data type



**- Domain Types in SQL**

char(n): 고정된 길이의 string.

varchar(n): 가변 길이의 string. N은 최대 길이

int: 정수, smallint: 작은 정수

numeric(p, d): Fixed point number. p가 유효숫자 자릿수, d가 소수점 이하가 몇 개인지.

총 p자리의 숫자인데 d가 소수점의 개수.

real, double precision: floating point와 double-precision floating point 넘버들.

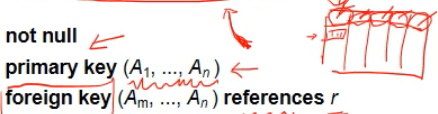
float(n): floating point number. 최소 n개의 digits의 정확도. 즉 최소 n개의 bit 수.

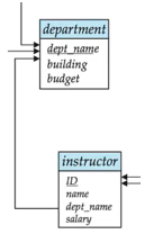
**모든 도메인 타입에 null 값은 허용됨.**

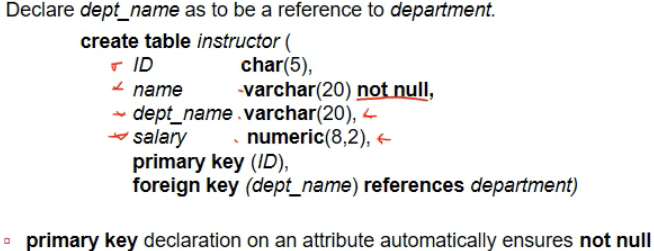
**- Integrity Constraints in Create Table**

데이터 타입을 정의하는 것 자체가 integrity constraints를 정의하는 것과 같다.

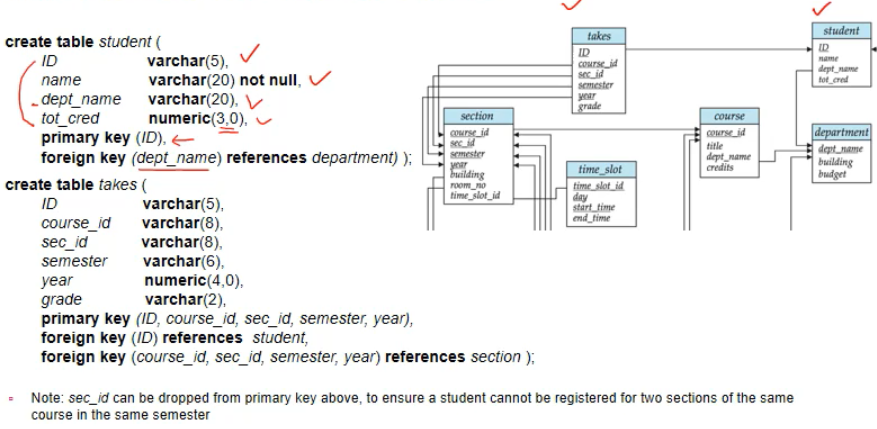
Integrity Constraints: 테이블의 value가 아무 값이나 들어오는 게 아니라 지켜야 할 특정한 조건을 제시하는 것.



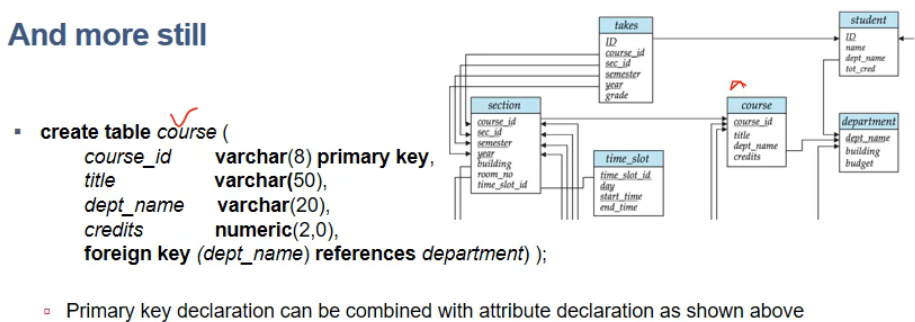
이 테이블에 대해 아래와 같은 조건들이 생김.



**Primary key는 자동으로 not null, uniqueness를 보증함**

****

sec\_id 에서 primary key 제약 조건을 빼면 같은 강좌의 001, 002 같은 섹션의 구분이 되지 않으므로 같은 학기에 같은 과목의 다른 섹션을 수강하지 못함. (중복 데이터가 됨)



위와 같이 single attribute가 primary key이면 attribute 오른쪽에 붙일 수도 있음.

- *Drop* and *Alter Table* Constructs

drop table: 데이터베이스에서 드랍된 relation에 대한 모든 정보 삭제.

문법: drop table TABLE\_NAME

alter table: 존재하는 relation에 attribute를 추가하거나 삭제.

문법: alter table r add A D

여기서 A는 attribute 이름, D는 A의 도메인.

Null이 새 attribute의 각각 튜플에 할당됨.

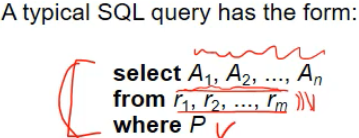
alter table r drop A

A는 relation r의 attribute 이름.

Attribute를 drop하는 것은 많은 데이터베이스들에서 지원하지 않는다.

**- Basic Structure of SQL Quries**

SQL은 relational operations 에 기반함. 수정과 발전이 있었음. Relational calculus에 기반.



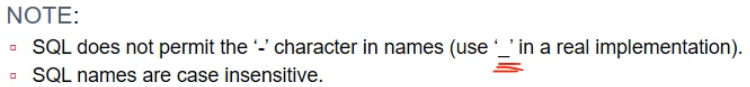
SQL 쿼리의 결과는 relation이 된다.

**- The *select* Clause**

Relational algebra의 select와는 무관. 독립적임.



모든 attribute를 select하려면 asterisk(\*) 사용.



SQL은 쿼리 결과와 relation에서 중복을 허용함.

중복을 없애기 위해서는 select 키워드 뒤에 **distinct** 라는 키워드를 붙임.

All은 중복 포함인데 기본값임. 즉 all은 안 써도 됨.

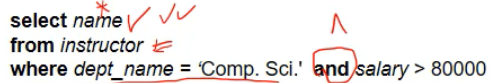
Select는 arithmetic expression을 지원함. => +, -, \*, /

상수나, attribute에 쓸 수 있음.



**- The *where* Clause**

Relational algebra의 selection predicate이랑 관계 있음.



비교 연산자: >, <, =, <=, >=, !=

Logical connectives: and, or, not

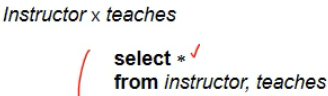


select에서 where절은 record by record로 비교. 다른 테이블이나 set끼리 비교가 안 됨. 그러니까 하고 싶으면 cartesian product를 해서 같은 record에서 비교해야 함.

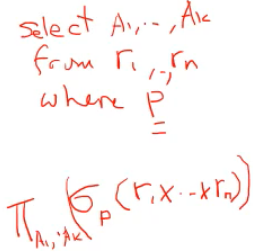
**- The *from* Clause**

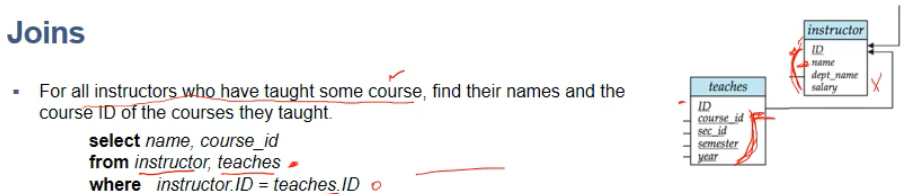
Expression의 계산에서 scan되어야 하는 relation들의 목록.

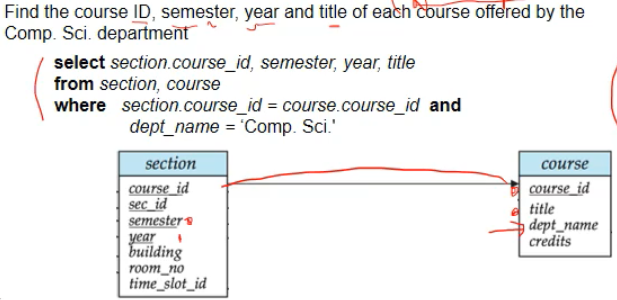
Relational algebra의 cartesian product와 관계 있음.



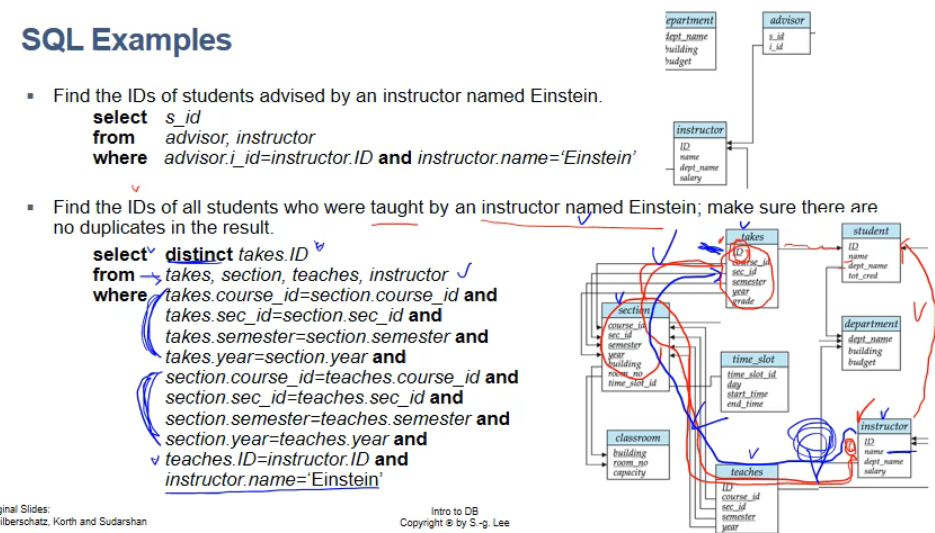
**- Joins**

 <- Select from where은 relational algebra로 왼쪽과 같음.





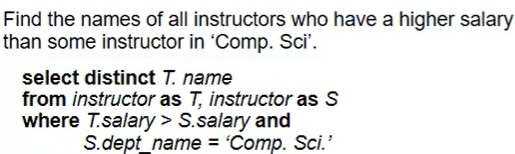
**- SQL Examples**



여기서는 section join 없이 해도 됨. Takes에서 section의 모든 정보를 담고 있기 때문.

**- The Rename Operation**

 <- as 사용.



Rename은 Alias (별명)로 부름. 근데 Tuple Variable로 이해해야 함. 단순히 이름만 바꾼 게 아니기 때문. (where 절에서 사용하는 걸 생각해보면 됨.)

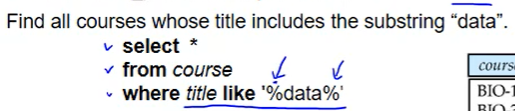
As는 안 써도 됨.

Select 절에서 rename할 수도 있고, from 절에서 rename 할 수도 있음. Update 절에서도 할 수 있다.

**- String Operations**

Character string들의 비교를 위함.

%는 모든 substring에 매칭, \_는 모든 character(1글자)에 매칭.

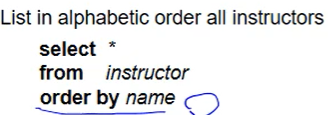


**스트링 와일드카드를 쓸 때는 항상 like를 써 줘야 함.**

=를 쓰면 와일드카드 기능을 하지 않고 문자열에서 “%”나 “\_”를 찾음.

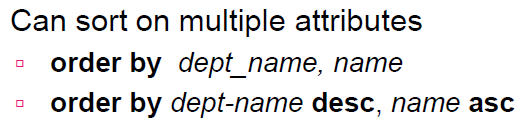
Like에서 그냥 문자 \_나 %를 찾고 싶을 때는 escape를 해줘야 함, “\%” 혹은 “\\_”.

**- Ordering the Display of Tuples**



Relation은 set이라서 순서가 의미 없었지만 여기선 실용적인 언어이기 때문에 순서가 정해지게 됨. DBMS에서는 순서가 의미가 없지만 retrieve 할 때 순서를 정해주는 것.

Order by name [desc|asc] 인데, asc가 default임.

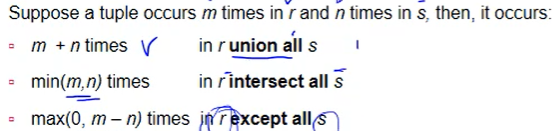


**- Set Operations**

Union, intersect, except가 존재함.

**각각의 set operation은 ‘자동으로 중복을 없앰’. 즉 default가 duplication 제거. 다른건 default가 duplication 놔두는 거. 왜냐면 set이기 때문.**

중복을 유지하고 싶으면 뒤에 all을 붙여야 함. (union all, intersect all, except all)



예시는 pdf 참조.

**- Null Values**

tuple들은 null 값을 가지는 것이 가능.

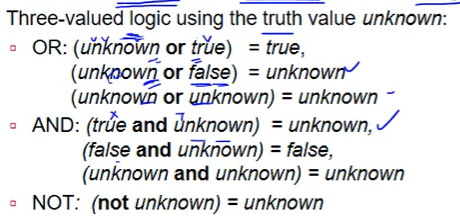
Null은 모르는 값이나 존재하지 않는 값으로 표시된다.

Null을 포함한 어떤 표현도 null이 됨. (ex: 5 + null = null)

Null을 체크할 때는 “is null”을 써야 함. ( = null 아님!) (= null 비교는 항상 false.)

왜냐?null은 three valued logic임. True, False, Unknown.

Null 과의 비교는 항상 unknown을 반환함. 이 때 unknown 계산은 밑의 식들처럼 되는데,



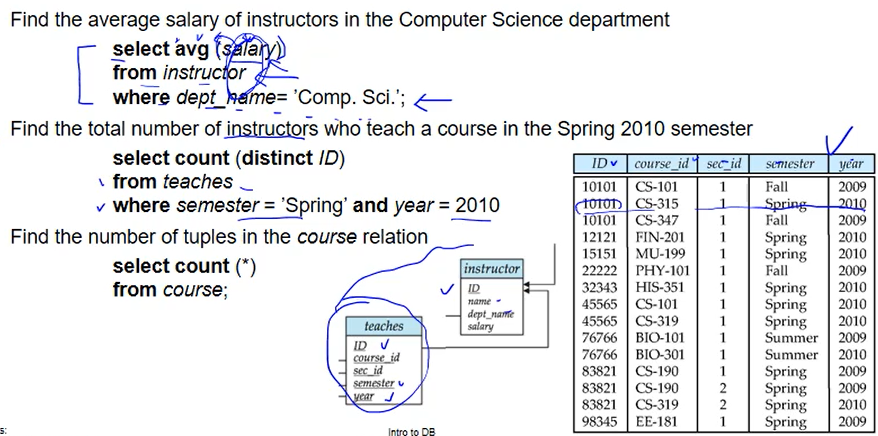


“P is unknown”은 P가unknown으로 계산되면 true를 반환함. 그리고unknown으로 계산이 되면 where 절의 값은 false로 취급된다. 따라서 a = null 같은 비교는 a is unknown이 되고 따라서 항상 false를 반환하게 됨.

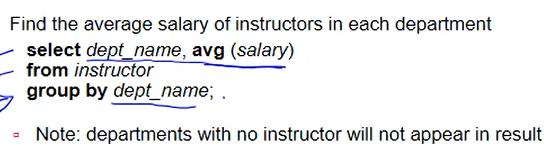
**- Aggregate Functions (집계 함수)**

여러 값을 받아서 하나의 값으로 표현. 프로그래밍에서 reduce와 같은 기능.

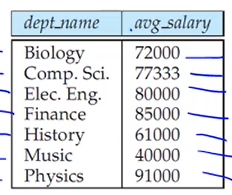
Avg, min, max, sum, count가 있음.



아래와 같이 그룹별로 avg를 하는 것도 가능.



이 경우 결과는 아래와 같이 나옴.



Group by를 쓸 경우, select 절에 group by에 있는 이름만 나와야 함. 다른 attribute가 나오면 에러.

**- Null Values and Aggregates**

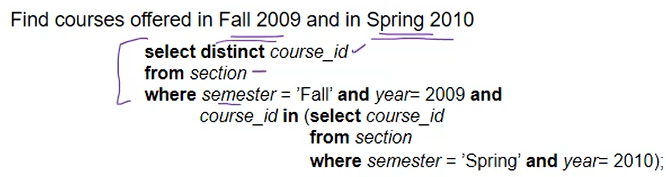
Aggregate 함수들은 null은 무시. 모든 value가 null이면 null 리턴.

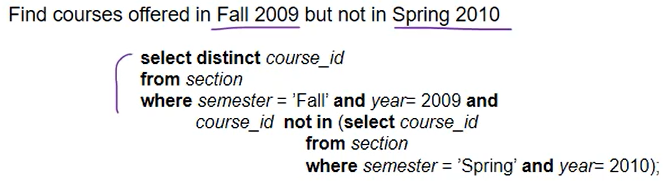
**Count(\*)는 null을 무시하지 않음.**

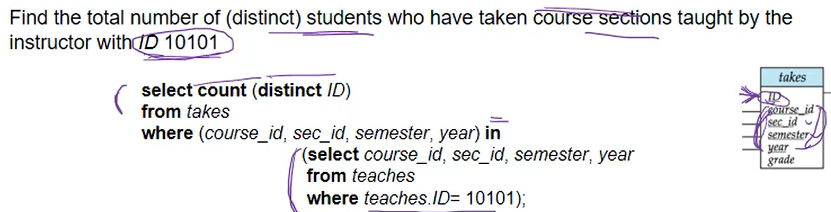
**- Nested Subqueries**

Select-from-where 표현이 다른 query에 중첩되는 것.

**- Nested Subqueries – Examples**

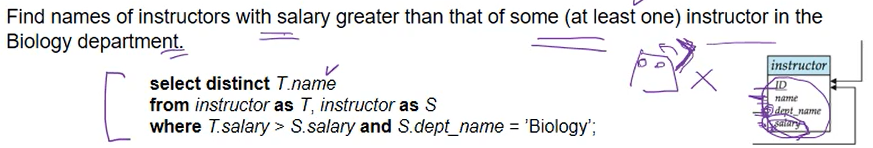




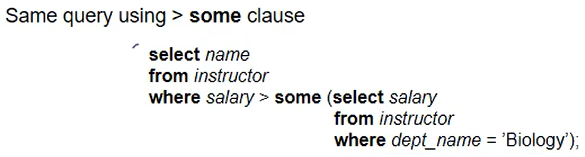


위에서 tuple constructor 사용. 괄호로 tuple들을 묶어 하나의 tuple처럼 표현하는 것. Python에서 tuple이랑 같은 형식으로 보면 됨.

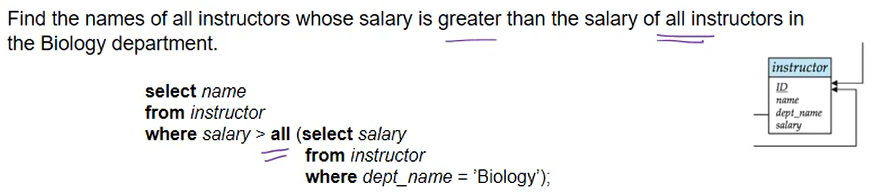
**- Set Comparison**



아래와 같이 some이라는 키워드를 사용해서 표현 가능.



all이라는 키워드도 있음.

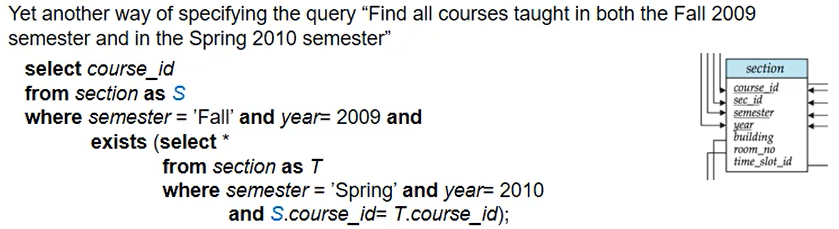


이 때 where절은 record by record로 나타내어지기 때문에, =some(~~) 이라고 하면 모든 레코드 중에 하나만 같으면 되고, =all(~~)하면 모든 레코드와 같아야 하는 거임.

**- Test for Empty Relations**

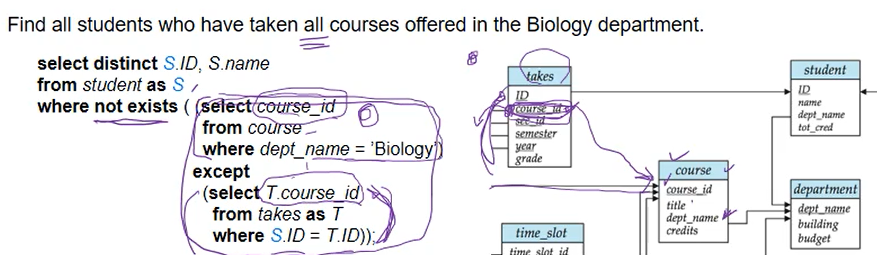
Exists는 subquery가 nonempty면 true를 반환. Not exists는 empty면 true.

**- Correlation Variables**



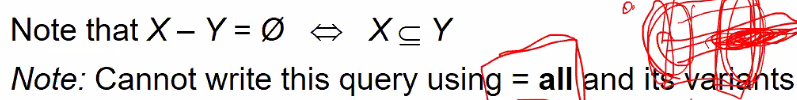
이 때 S를 써주는게 중요한게 exists는 공집합인지 아닌지를 보는 거니까 S를 사용해서 조건을 만들어줘서 의미있는 쿼리를 만들어 준다. 이 때 select를 as에 붙여서 비교하는 건 안된다고 하심. 예를 들어 select course\_id as C, C = T.course\_id 이게 안 된다는 뜻.

**- Not Exists**



except문의 뜻은 학생이 들은 모든 course\_id를 구하는 거임. 따라서

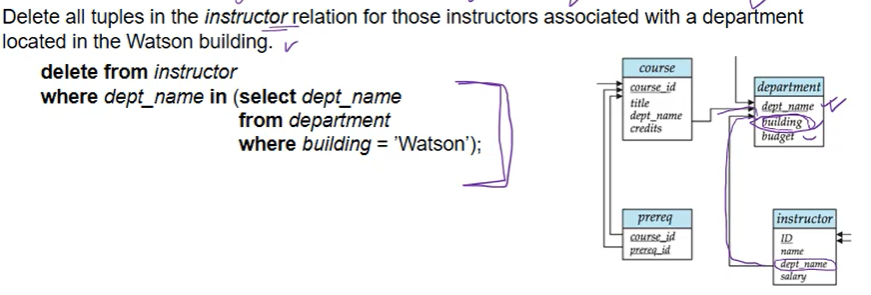
biology의 course\_id를 모두 찾아서, except에서 구한 걸 빼면 biology의 course\_id 중 학생이 듣지 않은 course가 나옴. 이게 not exists면 공집합이라는 소리니까 학생이 biology 학부의 모든 course를 들었다는 얘기.



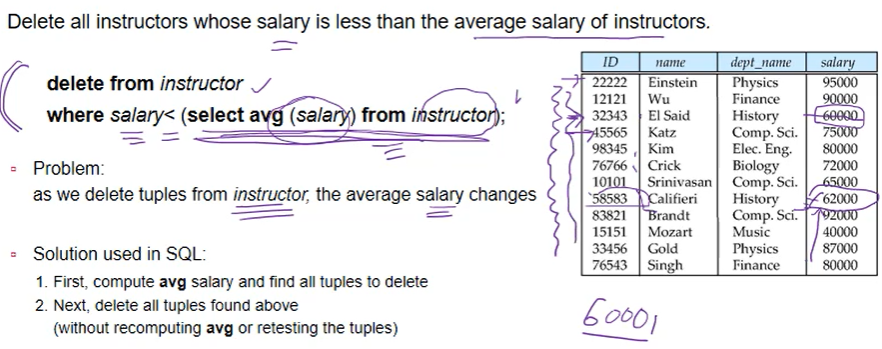
All을 사용하지 못하는 이유는 = all은 하나의 record가 비교문의 모든(all) record와 같은 지를 비교하는 키워드 때문에.

**- Modification of the Database – Deletion**

문법: delete from TABLE\_NAME where 조건;



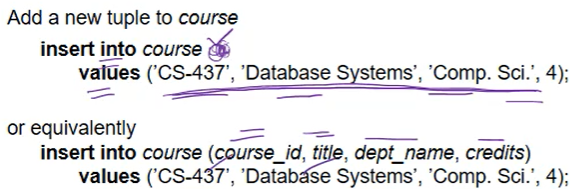
이런 위험한 연산에 delete는 경고해주지 않음!



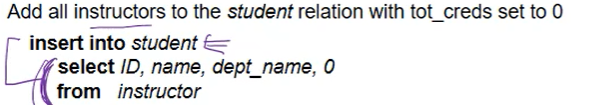
이렇게 avg가 계속 달라질 수 있기 때문에SQL에서는 미리 avg를 구한 다음에 그 값을 재사용.

**- Modification of the Database – Insertion**

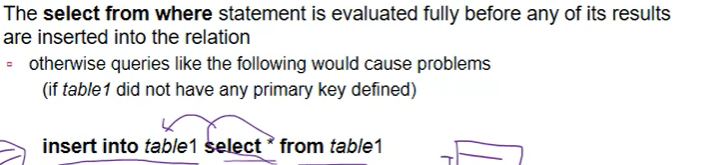
문법: insert into TABLE\_NAME values (…). 아래 두 개는 같음.



Insert에 select 문의 결과를 넣을 수 있음.

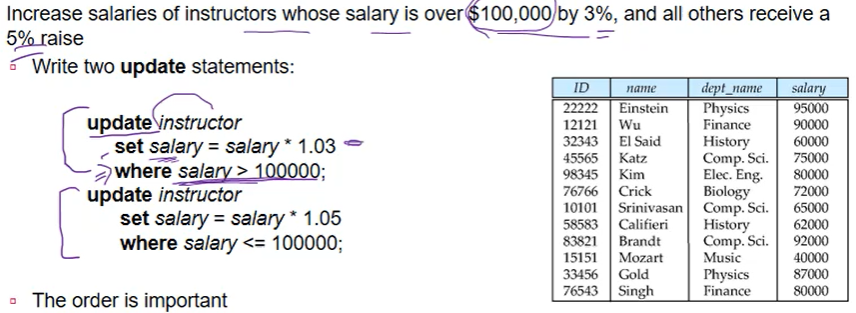


Delete와 같이 insert도subquery를 미리 구해놓고 그 다음 수행함. 문제 예방을 위해.



**- Modification of the Database – Updates**

문법: update TABLE\_NAME set … where …



순서가 중요함. Update 순서가 바뀌어서 수행되면 결과가 달라지기 때문에.