


# Aggregates

Other common aggregates:  
**max, min, sum, count, stdev, ...**

```
select count (distinct ID)  
from teaches  
where semester = ' Spring' and year = 2010
```




Find the average salary of instructors  
in the Computer Science

```
select avg(salary)  
from instructor  
where dept_name = 'Comp. Sci';
```

Can specify aggregates in any query.

Find max salary over instructors teaching in S'10


```
select max(salary)  
from teaches natural join instructor  
where semester = ' Spring' and year = 2010;
```



Aggregate result can be used as a scalar.

Find instructors with max salary:

```
select *  
from instructor  
where salary = (select max(salary) from instructor);
```



# Aggregates

Aggregate result can be used as a scalar.

Find instructors with max salary:

```
select *  
from instructor  
where salary = (select max(salary) from instructor);
```

Following doesn't work:

```
select *  
from instructor  
where salary = max(salary);
```

```
select name, max(salary)  
from instructor  
where salary = max(salary);
```

# Aggregates: Group By

Split the tuples into groups, and computer the aggregate for each group

```
select dept_name, avg (salary)
```

```
from instructor
```

```
group by dept_name;
```

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000

<i>dept_name</i>	<i>avg_salary</i>
Biology	72000
Comp. Sci.	77333
Elec. Eng.	80000
Finance	85000
History	61000
Music	40000
Physics	91000

# Aggregates: Group By

Attributes in the select clause **must be aggregates**, or **must appear in the group by** clause. Following wouldn't work

```
select dept_name, ID, avg (salary)
from instructor
group by dept_name;
```

“having” can be used to select only some of the groups.

```
select dept_name, avg (salary)
from instructor
group by dept_name
having avg(salary) > 42000;
```

# Aggregates and NULLs

Given

branch =

<u>bname</u>	<u>bcity</u>	<u>assets</u>
Downtown	Boston	9M
Perry	Horseneck	1.7M
Mianus	Horseneck	.4M
Waltham	Boston	NULL

## Aggregate Operations

```
SELECT SUM (assets) =  
FROM branch
```

<u>SUM</u>
11.1 M

*NULL is ignored for SUM*

*Same for AVG (3.7M), MIN (0.4M),  
MAX (9M)*

Also for COUNT(assets) -- returns 3

*But COUNT ( \* ) returns*

<u>COUNT</u>
4

# Aggregates and NULLs

Given

branch =

<u>bname</u>	<u>bcity</u>	<u>assets</u>
--------------	--------------	---------------

SELECT SUM (assets) =  
FROM branch

<u>SUM</u>
NULL

- *Same as AVG, MIN, MAX*
- *But COUNT (assets) returns*

<u>COUNT</u>
0

# Joins revisited

- Relation *instructor1*

<i>ID</i>	<i>name</i>	<i>dept_name</i>
10101 12121 15151	Srinivasan Wu Mozart	Comp. Sci. Finance Music

- Relation *teaches1*

<i>ID</i>	<i>course_id</i>
10101 12121 76766	CS-101 FIN-201 BIO-101

# Outer Join

- An extension of the join operation that avoids loss of information.
- Computes the join and then adds tuples from one relation that does not match tuples in the other relation to the result of the join.
- Uses *null* values:
  - *null* signifies that the value is unknown or does not exist



# Outer Join – Example

- Join

*instructor* ⋈ *teaches*

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>course_id</i>
10101 12121	Srinivasan Wu	Comp. Sci. Finance	CS-101 FIN-201

Left Outer Join

*instructor* ⋈<sub>L</sub> *teaches*

*In SQL: select \* from instructor natural left outer join teaches*

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>course_id</i>
10101 12121 15151	Srinivasan Wu Mozart	Comp. Sci. Finance Music	CS-101 FIN-201 <i>null</i>

# Outer Join – Example

## Right Outer Join

*instructor* ⋈<sub>⊂</sub> *teaches*

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>course_id</i>
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201
76766	null	null	BIO-101

## Full Outer Join

*instructor* ⋈<sub>⊃</sub> *teaches*

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>course_id</i>
10101	Srinivasan	Comp. Sci.	CS-101
12121	Wu	Finance	FIN-201
15151	Mozart	Music	<i>null</i>
76766	null	null	BIO-101

# Nested queries

- ▶ At the end of class we attempted to create a query that returns (from the university dataset) the building that houses the departments with the largest average salary.
- ▶ We started with the following, illegal query:
  - `select building, max(avg(budget)) from department group by building;`
- ▶ We ended class with:
  - `select building, avg_budget  
from  
 (select building, avg(budget) as avg_budget from department group by building)  
 as avg_budgets  
where avg_budget = (select (max(avg_budget) from  
 (select building, avg(budget) as avg_budget from department group by building));`
- ▶ There were only two more things that needed to be fixed from the query above:
  - The extra parenthesis before the word max should be removed
  - Nested queries in the from clause need to be given a name even if you don't use it (so we need to name the final nested query via adding "as avg\_budgets" like we did for the first one).
- ▶ So the final correct query is:  
`select building, avg_budget  
from  
 (select building, avg(budget) as avg_budget from department group by building)  
 as avg_budgets  
where avg_budget = (select max(avg_budget) from  
 (select building, avg(budget) as avg_budget from department group by building)  
 as avg_budgets);`