

# CSE 344: Intro to Data Management

## Aggregates

Paul G. Allen School of Computer Science and Engineering  
University of Washington, Seattle

# Aggregates

- Aggregate: many values to one value

# Aggregates

- Aggregate: many values to one value
  
- Aggregates in SQL:
  - $\text{sum}(1, 4, 3, 4) = 1+4+3+4 = 12$

# Aggregates

- Aggregate: many values to one value
  
- Aggregates in SQL:
  - $\text{sum}(1, 4, 3, 4) = 1+4+3+4 = 12$
  - $\text{max}(1, 4, 3, 4) = 4$
  - $\text{min}(1, 4, 3, 4) = 1$

# Aggregates

- Aggregate: many values to one value
  
- Aggregates in SQL:
  - $\text{sum}(1, 4, 3, 4) = 1+4+3+4 = 12$
  - $\text{max}(1, 4, 3, 4) = 4$
  - $\text{min}(1, 4, 3, 4) = 1$
  - $\text{count}(1, 4, 3, 4) = 4$

# Aggregates

- Aggregate: many values to one value
  
- Aggregates in SQL:
  - $\text{sum}(1, 4, 3, 4) = 1+4+3+4 = 12$
  - $\text{max}(1, 4, 3, 4) = 4$
  - $\text{min}(1, 4, 3, 4) = 1$
  - $\text{count}(1, 4, 3, 4) = 4$
  - $\text{avg}(1, 4, 3, 4) = 3$

# Aggregates

- Aggregate: many values to one value
  
- Aggregates in SQL:
  - $\text{sum}(1, 4, 3, 4) = 1+4+3+4 = 12$
  - $\text{max}(1, 4, 3, 4) = 4$
  - $\text{min}(1, 4, 3, 4) = 1$
  - $\text{count}(1, 4, 3, 4) = 4$
  - $\text{avg}(1, 4, 3, 4) = 3$



The collection may have duplicates!

# COUNT

How many records are in payroll?

```
SELECT count(*) as C  
FROM payroll;
```

C

4

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

**regist**

user_id	car
123	Charger
567	Civic
567	Pinto



# COUNT

How many records are in payroll?

```
SELECT count (*)  
FROM payroll;
```

```
count(*)
```

```
4
```

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

**regist**

user_id	car
123	Charger
567	Civic
567	Pinto

# COUNT

How many records are in payroll?

```
SELECT count (*)  
FROM payroll;
```

count(\*)

4

How many cars are in the database?

```
SELECT count (*)  
FROM regist;
```

...

3

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

**regist**

user_id	car
123	Charger
567	Civic
567	Pinto

# COUNT

How many records are in payroll?

```
SELECT count (*)  
FROM payroll;
```

count(\*)

4

How many cars are in the database?

```
SELECT count (*)  
FROM regist;
```

...

3

How many TA's are there?

```
SELECT count (*)  
FROM payroll  
WHERE job='TA';
```

...

2

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

# COUNT

How many records are in payroll?

```
SELECT count (*)  
FROM payroll;
```

count(\*)

4

How many cars are in the database?

```
SELECT count (*)  
FROM regist;
```

...

3

How many TA's are there?

```
SELECT count (*)  
FROM payroll  
WHERE job='TA';
```

...

2

How many people have salary > 55000?

```
SELECT count (*)  
FROM payroll  
WHERE salary>55000;
```

...

3

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

# SUM, MIN, MAX, AVG

What is the sum of all salaries?

```
SELECT sum(salary)
FROM payroll;
```

sum(...)

300000

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

**regist**

user_id	car
123	Charger
567	Civic
567	Pinto

# SUM, MIN, MAX, AVG

What is the sum of all salaries?

```
SELECT sum(salary)
FROM payroll;
```

sum(...)

300000

What is the average salary?

```
SELECT avg(salary)
FROM payroll;
```

avg(...)

75000

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

**regist**

user_id	car
123	Charger
567	Civic
567	Pinto

# SUM, MIN, MAX, AVG

What is the sum of all salaries?

```
SELECT sum(salary)
FROM payroll;
```

sum(...)

300000

What is the average salary?

```
SELECT avg(salary)
FROM payroll;
```

avg(...)

75000

What is the smallest salary?  
What is the largest salary?



On your own

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

**regist**

user_id	car
123	Charger
567	Civic
567	Pinto

```
SELECT agg(attrs)  
FROM ... WHERE ...;
```



# Semantics

count or  
sum or ...

```
SELECT agg (attrs)  
FROM ... WHERE ...;
```

# Semantics

count or  
sum or ...

```
SELECT agg (attrs)  
FROM ... WHERE ...;
```

\* or salary or ...

# Semantics

count or  
sum or ...

```
SELECT agg(attrs)  
FROM ... WHERE ...;
```

\* or salary or ...

Step 1:  
drop aggregate,  
compute query



```
SELECT attrs  
FROM ... WHERE ...;
```

# Semantics

count or  
sum or ...

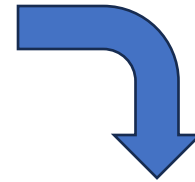
```
SELECT agg(attrs)  
FROM ... WHERE ...;
```

\* or salary or ...

Step 1:  
drop aggregate,  
compute query



```
SELECT attrs  
FROM ... WHERE ...;
```



attrs	...

# Semantics

count or  
sum or ...

```
SELECT agg(attrs)  
FROM ... WHERE ...;
```

\* or salary or ...

Step 1:  
drop aggregate,  
compute query

```
SELECT attrs  
FROM ... WHERE ...;
```

Step 2:  
apply aggregate

agg
55

attrs	...

# COUNT

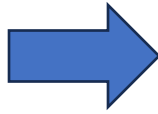
```
SELECT count(*)  
FROM payroll;
```

## payroll

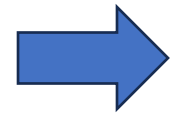
user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# COUNT

```
SELECT count(*)  
FROM payroll;
```



```
SELECT *  
FROM payroll;
```



...

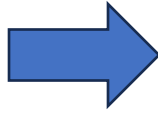
4

## payroll

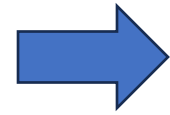
user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# COUNT

```
SELECT count(*)  
FROM payroll;
```



```
SELECT *  
FROM payroll;
```



...

4

How many jobs are there in this institution?

```
SELECT count(job)  
FROM payroll;
```

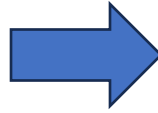
## payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

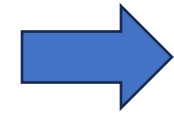


# COUNT

```
SELECT count (*)  
FROM payroll;
```



```
SELECT *  
FROM payroll;
```



```
...  
4
```

How many jobs are there in this institution?

```
SELECT count (job)  
FROM payroll;
```



```
SELECT job  
FROM payroll;
```

?

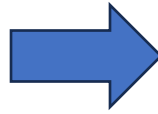


**payroll**

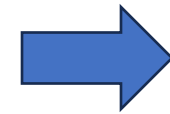
user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# COUNT

```
SELECT count(*)  
FROM payroll;
```



```
SELECT *  
FROM payroll;
```



```
...  
4
```

How many jobs are there in this institution?

```
SELECT count(job)  
FROM payroll;
```



```
SELECT job  
FROM payroll;
```



job

TA

TA

Prof

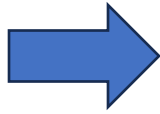
Prof

payroll

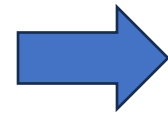
user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# COUNT

```
SELECT count(*)  
FROM payroll;
```



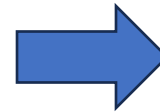
```
SELECT *  
FROM payroll;
```



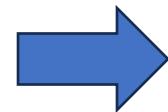
...
4

How many jobs are there in this institution?

```
SELECT count(job)  
FROM payroll;
```



job
TA
TA
Prof
Prof



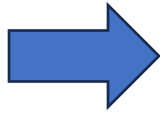
...
4

## payroll

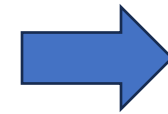
user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# COUNT

```
SELECT count(*)  
FROM payroll;
```



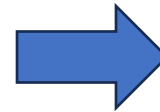
```
SELECT *  
FROM payroll;
```



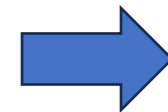
...
4

How many jobs are there in this institution?

```
SELECT count(job)  
FROM payroll;
```



job
TA
TA
Prof
Prof



...
4

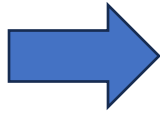
**WRONG!**

## payroll

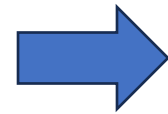
user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# COUNT

```
SELECT count(*)  
FROM payroll;
```



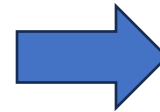
```
SELECT *  
FROM payroll;
```



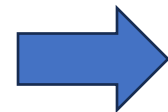
...
4

How many jobs are there in this institution?

```
SELECT count(job)  
FROM payroll;
```



job
TA
TA
Prof
Prof



...
4

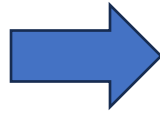
```
SELECT count(DISTINCT job)  
FROM payroll;
```

**payroll**

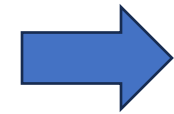
user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# COUNT

```
SELECT count(*)  
FROM payroll;
```



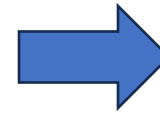
```
SELECT *  
FROM payroll;
```



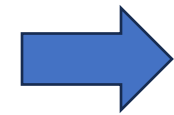
```
...  
4
```

How many jobs are there in this institution?

```
SELECT count(job)  
FROM payroll;
```



```
job  
TA  
TA  
Prof  
Prof
```



```
...  
4
```

```
SELECT count(DISTINCT job)  
FROM payroll;
```



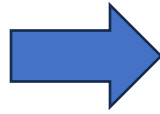
```
job  
TA  
Prof
```

payroll

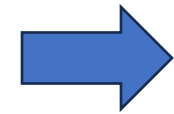
user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# COUNT

```
SELECT count(*)  
FROM payroll;
```



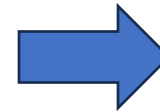
```
SELECT *  
FROM payroll;
```



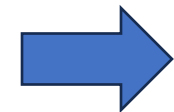
```
...  
4
```

How many jobs are there in this institution?

```
SELECT count(job)  
FROM payroll;
```



```
job  
TA  
TA  
Prof  
Prof
```

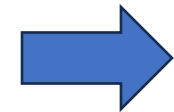


```
...  
4
```

```
SELECT count(DISTINCT job)  
FROM payroll;
```



```
job  
TA  
Prof
```



```
...  
2
```



payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# Aggregates and NULLs

Aggregates ignore NULLs:

- Sum: same as 0
- Avg: NOT the same as 0
- Min/max: same as  $+\infty$ ,  $-\infty$
- count: doesn't include them, but it's more subtle



# Aggregates and NULLs

```
SELECT sum(salary)  
FROM payroll;
```

sum(...)

200000

50000 + 60000 + 90000

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	NULL	60000
567	Magda	Prof	90000
789	Dan	Prof	NULL
432	NULL	Prof	NULL

# Aggregates and NULLs

```
SELECT sum(salary)
FROM payroll;
```

sum(...)

200000

50000 + 60000 + 90000

```
SELECT avg(salary)
FROM payroll;
```

avg(...)

66667

NULLs are just ignored.  
Just as you expected.

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	NULL	60000
567	Magda	Prof	90000
789	Dan	Prof	NULL
432	NULL	Prof	NULL

# Discussion: Aggregates

- Semantics: two steps
- NULLs are ignored

# Aggregates and Joins

# Aggregates and Joins

- Joins combine records from multiple tables
- Aggregates: many values to one value
- Together they form a very powerful SQL tool

# Aggregates and Joins

Find the average salary of people driving a Pinto

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

**regist**

user_id	car
123	Charger
123	Pinto
567	Civic
567	Pinto

# Aggregates and Joins

Find the average salary of people driving a Pinto

```
SELECT avg(P.salary)
FROM payroll P, regist R
WHERE P.user_id = R.user_id
       and R.car = 'Pinto';
```

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

**regist**

user_id	car
123	Charger
123	Pinto
567	Civic
567	Pinto

# Aggregates and Joins

Find the average salary of people driving a Pinto

```
SELECT avg(P.salary)
FROM payroll P, regist R
WHERE P.user_id = R.user_id
and R.car = 'Pinto';
```

```
SELECT P.salary
FROM payroll P, regist R
...;
```



name	salary
Jack	50000
Magda	90000

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
123	Pinto
567	Civic
567	Pinto



# Aggregates and Joins

Find the average salary of people driving a Pinto

```
SELECT avg(P.salary)
FROM payroll P, regist R
WHERE P.user_id = R.user_id
and R.car = 'Pinto';
```

```
SELECT P.salary
FROM payroll P, regist R
...;
```

A blue arrow points from the SQL query box to a table with two columns: 'name' and 'salary'. The 'salary' column is highlighted in blue. The table contains two rows: Jack with salary 50000 and Magda with salary 90000. A second blue arrow points from this table to another table with two rows: 'avg(...)' and '70000'.

name	salary
Jack	50000
Magda	90000

avg(...)
70000

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
123	Pinto
567	Civic
567	Pinto

# Duplicates

- Need to watch for duplicates introduced when we join two tables
- Sometimes duplicates are easy to deal with, e.g. `COUNT(DISTINCT ...)`
- Sometimes they are much harder to deal with, and we will discuss this in future lectures

# Duplicates

How many people drive a car?

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

# Duplicates

How many people drive a car?

```
SELECT count(*)  
FROM payroll P, regist R  
WHERE P.user_id = R.user_id;
```

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

# Duplicates

How many people drive a car?

```
SELECT count(*)  
FROM payroll P, regist R  
WHERE P.user_id = R.user_id;
```

```
SELECT *  
FROM ...;
```



user_id	name	job	salary	user_id	car
123	Jack	TA	50000	123	Charger
567	Magda	Prof	90000	567	Civic
567	Magda	Prof	90000	567	Pinto

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

# Duplicates

How many people drive a car?

```
SELECT count(*)  
FROM payroll P, regist R  
WHERE P.user_id = R.user_id;
```

```
SELECT *  
FROM ...;
```



user_id	name	job	salary	user_id	car
123	Jack	TA	50000	123	Charger
567	Magda	Prof	90000	567	Civic
567	Magda	Prof	90000	567	Pinto

count(*)
3

Wrong!



payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

# Duplicates

How many people drive a car?

```
SELECT count(DISTINCT user_id)
FROM payroll P, regist R
WHERE P.user_id = R.user_id;
```

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

# Duplicates

How many people drive a car?

```
SELECT count(DISTINCT user_id)
FROM payroll P, regist R
WHERE P.user_id = R.user_id;
```

```
SELECT DISTINCT user_id
FROM ...;
```



user_id
123
567

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto



# Duplicates

How many people drive a car?

```
SELECT count(DISTINCT user_id)
FROM payroll P, regist R
WHERE P.user_id = R.user_id;
```

```
SELECT DISTINCT user_id
FROM ...;
```



user_id
123
567



Right!

count(\*)

2

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

# Duplicates

What is the average salary of car drivers?

```
SELECT avg(P.salary)
FROM payroll P, regist R
WHERE P.user_id = R.user_id;
```

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

# Duplicates

What is the average salary of car drivers?

```
SELECT avg(P.salary)
FROM payroll P, regist R
WHERE P.user_id = R.user_id;
```

```
SELECT P.salary
FROM ...;
```



name	salary
Jack	50000
Magda	90000
Magda	90000

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

# Duplicates

What is the average salary of car drivers?

```
SELECT avg(P.salary)
FROM payroll P, regist R
WHERE P.user_id = R.user_id;
```

avg(...)
76667

```
SELECT P.salary
FROM ...;
```



name	salary
Jack	50000
Magda	90000
Magda	90000



payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

# Duplicates

What is the average salary of car drivers?

```
SELECT avg(P.salary)
FROM payroll P, regist R
WHERE P.user_id = R.user_id;
```

```
SELECT P.salary
FROM ...;
```



name	salary
Jack	50000
Magda	90000
Magda	90000



avg(...)
76667



payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

# Duplicates

What is the average salary of car drivers?

```
SELECT avg(DISTINCT P.salary)
FROM payroll P, regist R
WHERE P.user_id = R.user_id;
```

Does DISTINCT fix it?

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
567	Civic
567	Pinto

# Duplicates

What is the average salary of car drivers?

```
SELECT avg(DISTINCT P.salary)
FROM payroll P, regist R
WHERE P.user_id = R.user_id;
```

```
SELECT DISTINCT P.salary
FROM ...;
```

Does DISTINCT fix it?

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	50000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
345	Tesla
567	Civic
567	Pinto

# Duplicates

What is the average salary of car drivers?

```
SELECT avg(DISTINCT P.salary)
FROM payroll P, regist R
WHERE P.user_id = R.user_id;
```

```
SELECT DISTINCT P.salary
FROM ...;
```



salary
50000
90000



avg(...)

70000

Wrong!

Does DISTINCT fix it?

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	50000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
345	Tesla
567	Civic
567	Pinto



# Duplicates

What is the average salary of car drivers?

```
SELECT avg(DISTINCT P.salary)
FROM payroll P, regist
WHERE P.user_id = R.
```

```
SELECT DISTINCT P.salary
FROM ...;
```

This query is harder to fix. We will discuss it later



salary
50000
90000



avg(...)
70000

Wrong!

Does DISTINCT fix it?

Correct answer: 63333

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	50000
567	Magda	Prof	90000
789	Dan	Prof	100000

regist

user_id	car
123	Charger
345	Tesla
567	Civic
567	Pinto

# Summary so far

- Aggregates
  - sum, min, max, count, avg
  - Two steps semantics
  - Subtle interactions with joins, duplicates, nulls

# Aggregates

```
SELECT count(*) as C  
FROM payroll  
WHERE job = 'TA';
```

May use alias



C
2

## payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# Aggregates

```
SELECT count(*) as C  
FROM payroll  
WHERE job = 'TA';
```

May use alias



C
2

```
SELECT count(*) as C, avg(salary) as A  
FROM payroll  
WHERE job = 'TA';
```



C	A
2	55000

## payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

We may compute several aggregates

# GROUP BY

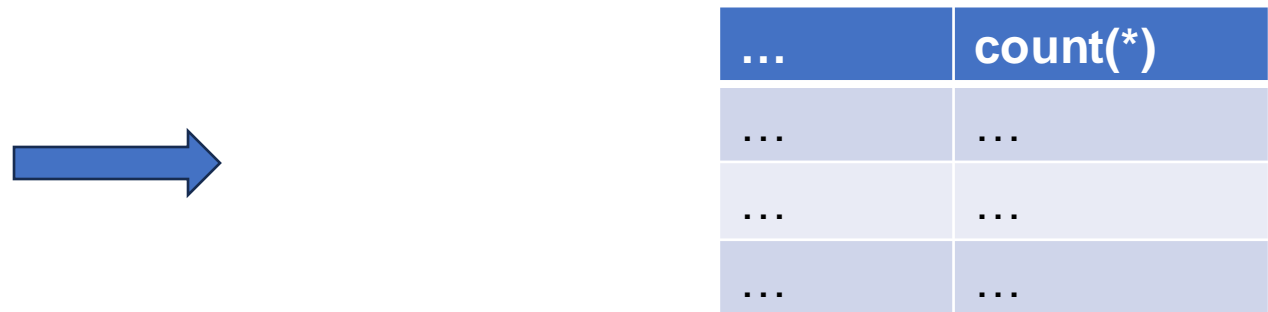
# Group By

- So far, a single aggregate, or a tuple of aggregates



count(*)	avg(salary)	count(distinct job)
...	...	...

- Next: compute a set of aggregates, one per group:



...	count(*)
...	...
...	...
...	...

# Group By Basics

```
SELECT job, avg(salary)
FROM payroll
GROUP BY job;
```

**payroll**

<b>user_id</b>	<b>name</b>	<b>job</b>	<b>salary</b>
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# Group By Basics

```
SELECT job, avg(salary)
FROM payroll
GROUP BY job;
```

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000



# Group By Basics

```
SELECT job, avg(salary)
FROM payroll
GROUP BY job;
```

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000



job	avg(salary)
TA	55000
Prof	95000

# Group By Basics

Find total revenue for each product.

## sales

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March

# Group By Basics

Find total revenue for each product.

```
SELECT product, sum(price*quant) as rev
FROM sales
GROUP BY product;
```

**sales**

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March

# Group By Basics

Find total revenue for each product.

```
SELECT product, sum(price*quant) as rev
FROM sales
GROUP BY product;
```

**sales**

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March

# Group By Basics

Find total revenue for each product.

```
SELECT product, sum(price*quant) as rev
FROM sales
GROUP BY product;
```

**sales**

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March

One row for each product



product	rev	
Bagel	140	60+50+30
Banana	75	25+50
Apple	40	40

# Group By Basics

Find total revenue for each **month**.

## sales

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March

# Group By Basics

Find total revenue for each **month**.

```
SELECT month, sum(price*quant) as rev
FROM sales
GROUP BY month;
```

## sales

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March

# Group By Basics

Find total revenue for each **month**.

```
SELECT month, sum(price*quant) as rev
FROM sales
GROUP BY month;
```

## sales

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March



## GROUP BY month

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Banana	0.5	50	Feb
Banana	5	10	Feb
Bagel	1.50	20	March
Apple	4	10	March



# Group By Basics

Find total revenue for each **month**.

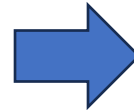
```
SELECT month, sum(price*quant) as rev
FROM sales
GROUP BY month;
```

## sales

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March

## GROUP BY **month**

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Banana	0.5	50	Feb
Banana	5	10	Feb
Bagel	1.50	20	March
Apple	4	10	March



# Group By Basics

Find total revenue for each **month**.

```
SELECT month, sum(price*quant) as rev
FROM sales
GROUP BY month;
```

One row for each month

month	rev	
Jan	140	60+50
Feb	75	25+50
March	40	40+30



**sales**

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March

**GROUP BY** month

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Banana	0.5	50	Feb
Banana	5	10	Feb
Bagel	1.50	20	March
Apple	4	10	March



# Group By Basics

Find total revenue per month, for sales over 2.50

```
SELECT month, sum(price*quant) as rev
FROM sales
WHERE price > 2.5
GROUP BY month;
```

## sales

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March

# Group By Basics

Find total revenue per month, for sales over 2.50

```
SELECT month, sum(price*quant) as rev
FROM sales
WHERE price > 2.5
GROUP BY month;
```

**sales**

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	<del>1.50</del>	20	March
Banana	<del>0.5</del>	50	Feb
Banana	5	10	Feb
Apple	4	10	March



Not interested  
in these sales

# Group By Basics

Find total revenue per month, for sales over 2.50

```
SELECT month, sum(price*quant) as rev
FROM sales
WHERE price > 2.5
GROUP BY month;
```

**sales**

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	<del>1.50</del>	20	March
Banana	<del>0.5</del>	50	Feb
Banana	5	10	Feb
Apple	4	10	March

**GROUP BY** month



product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Banana	5	10	Feb
Apple	4	10	March

# Group By Basics

Find total revenue per month, for sales over 2.50

```
SELECT month, sum(price*quant) as rev
FROM sales
WHERE price > 2.5
GROUP BY month;
```

One row for each month

month	rev	
Jan	140	60+50
Feb	75	25+50
March	40	40+30

sales

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	<del>1.50</del>	20	March
Banana	<del>0.5</del>	50	Feb
Banana	5	10	Feb
Apple	4	10	March

GROUP BY month

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Banana	5	10	Feb
Apple	4	10	March

# Group By Basics

Find total revenue for each product and each month.

```
SELECT product, month, sum(price*quant) as rev
FROM sales
GROUP BY product, month;
```

**sales**

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March

# Group By Basics

Find total revenue for each product and each month.

```
SELECT product, month, sum(price*quant) as rev
FROM sales
GROUP BY product, month;
```

**sales**

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March



product	month	rev
Bagel	Jan	110
Bagel	March	30
Banana	Feb	75
Apple	March	40



# A Source of Errors

What does this query return?

```
SELECT product, price, sum(price*quant) as rev
FROM sales
GROUP BY product;
```

**sales**

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March

# A Source of Errors

What does this query return?

```
SELECT product, price, sum(price*quant) as rev
FROM sales
GROUP BY product;
```

**sales**

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March

One row for each product

No unique price for the group



product	price	rev
Bagel	??	140
Banana	??	75
Apple	??	40

# A Source of Errors

What does this query return?

```
SELECT product, price, sum(price*quant) as rev  
FROM sales  
GROUP BY product;
```

Rule: every attribute in SELECT must also occur in GROUP BY

sales

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March

One row for each product

No unique price for the group

product	price	rev
Bagel	??	140
Banana	??	75
Apple	??	40

# Discussion so far

- **GROUP BY:** list of attributes
- **SELECT:** some group-by attrs, and aggregates
- One output tuple for each group

# Semantics

# Semantics

```
SELECT attr1, attr2, ..., agg1(..), agg2(..), ..  
FROM Tables  
WHERE Condition  
GROUP BY attr1, attr2, ..;
```

- Step 1: compute **SELECT \* FROM .. WHERE..**
- Step 2: **GROUP BY**
- Step 3: for each group emit 1 output

# Example

```
SELECT month, sum(quant)
FROM sales
WHERE price < 4.5
GROUP BY month;
```

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March

# Example

```
SELECT month, sum(quant)
FROM sales
WHERE price < 4.5
GROUP BY month;
```

## Step 1

```
SELECT *
FROM sales
WHERE price < 4.5;
```

product	price	quant	month
Bagel	3	20	Jan
<del>Bagel</del>	<del>5</del>	<del>40</del>	<del>Jan</del>
Bagel	1.50	20	March
Banana	0.5	50	Feb
<del>Banana</del>	<del>5</del>	<del>40</del>	<del>Feb</del>
Apple	4	10	March

product	price	quant	month
Bagel	3	20	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Apple	4	10	March



# Example

```
SELECT month, sum(quant)
FROM sales
WHERE price < 4.5
GROUP BY month;
```

## Step 1

```
SELECT *
FROM sales
WHERE price < 4.5;
```

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March

product	price	quant	month
Bagel	3	20	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Apple	4	10	March

## Step 2 Group-by

product	price	quant	month
Bagel	3	20	Jan
Banana	0.5	50	Feb
Bagel	1.50	20	March
Apple	4	10	March

# Example

```
SELECT month, sum(quant)
FROM sales
WHERE price < 4.5
GROUP BY month;
```

Each group, one output

## Step 1

```
SELECT *
FROM sales
WHERE price < 4.5;
```

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March

product	price	quant	month
Bagel	3	20	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Apple	4	10	March


## Step 2 Group-by

product	price	quant	month	month	quant
Bagel	3	20	Jan	Jan	20
Banana	0.5	50	Feb	Feb	50
Bagel	1.50	20	March	March	30
Apple	4	10	March		

## Step 3

# Multiple Aggregates

```
SELECT product, count(*), sum(quant)
FROM sales
GROUP BY product;
```




product	count...	sum...
Bagel	3	50
Banana	2	60
Apple	1	10

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March

# Multiple Aggregates

```
SELECT product, count(*), sum(quant)
FROM sales
GROUP BY product;
```




product	count...	sum...
Bagel	3	50
Banana	2	60
Apple	1	10

```
SELECT product, count(*)
FROM sales
GROUP BY product;
```

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March

# Multiple Aggregates

```
SELECT product, count(*), sum(quant)
FROM sales
GROUP BY product;
```



product	count...	sum...
Bagel	3	50
Banana	2	60
Apple	1	10

```
SELECT product, count(*)
FROM sales
GROUP BY product;
```




product	count...
Bagel	3
Banana	2
Apple	1

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March

# Multiple Aggregates

```
SELECT product, count(*), sum(quant)
FROM sales
GROUP BY product;
```



product	count...	sum...
Bagel	3	50
Banana	2	60
Apple	1	10

```
SELECT product, count(*)
FROM sales
GROUP BY product;
```




product	count...
Bagel	3
Banana	2
Apple	1

```
SELECT product
FROM sales
GROUP BY product;
```

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March

# Multiple Aggregates

```
SELECT product, count(*), sum(quant)
FROM sales
GROUP BY product;
```



product	count...	sum...
Bagel	3	50
Banana	2	60
Apple	1	10

```
SELECT product, count(*)
FROM sales
GROUP BY product;
```



product	count...
Bagel	3
Banana	2
Apple	1

```
SELECT product
FROM sales
GROUP BY product;
```



product
Bagel
Banana
Apple

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March

# Multiple Aggregates

```
SELECT product, count(*), sum(quant)
FROM sales
GROUP BY product;
```



product	count...	sum...
Bagel	3	50
Banana	2	60
Apple	1	10

```
SELECT product, count(*)
FROM sales
GROUP BY product;
```



product	count...
Bagel	3
Banana	2
Apple	1

```
SELECT product
FROM sales
GROUP BY product;
```



product
Bagel
Banana
Apple

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March

Same as

```
SELECT DISTINCT product
FROM sales;
```



# Coping with Empty Groups

# Coping with Empty Groups

- A group is never empty, by definition!
- Therefore  $\text{count}(\ast) \geq 1$
- Sometimes we want answers with  $\text{count}(\dots)=0$
- Then we use outer-joins

# Coping with Empty Groups

```
SELECT job, count(*)  
FROM payroll  
GROUP BY job;
```

job	count(*)
TA	2
Prof	2

count people  
per job

## payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# Coping with Empty Groups

```
SELECT job, count(*)  
FROM payroll  
GROUP BY job;
```

job	count(*)
TA	2
Prof	2

```
SELECT job, count(*)  
FROM payroll  
WHERE salary > 55000  
GROUP BY job;
```

## payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# Coping with Empty Groups

```
SELECT job, count(*)  
FROM payroll  
GROUP BY job;
```

job	count(*)
TA	2
Prof	2

```
SELECT job, count(*)  
FROM payroll  
WHERE salary > 55000  
GROUP BY job;
```

job	count(*)
TA	1
Prof	2

## payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# Coping with Empty Groups

```
SELECT job, count(*)  
FROM payroll  
GROUP BY job;
```

job	count(*)
TA	2
Prof	2

```
SELECT job, count(*)  
FROM payroll  
WHERE salary > 55000  
GROUP BY job;
```

job	count(*)
TA	1
Prof	2

```
SELECT job, count(*)  
FROM payroll  
WHERE salary > 75000  
GROUP BY job;
```

## payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# Coping with Empty Groups

```
SELECT job, count(*)  
FROM payroll  
GROUP BY job;
```

job	count(*)
TA	2
Prof	2

```
SELECT job, count(*)  
FROM payroll  
WHERE salary > 55000  
GROUP BY job;
```

job	count(*)
TA	1
Prof	2

```
SELECT job, count(*)  
FROM payroll  
WHERE salary > 75000  
GROUP BY job;
```

TA group  
no longer  
exists

job	count(*)
Prof	2

## payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# Coping with Empty Groups

```
SELECT job, count(*)  
FROM payroll  
GROUP BY job;
```

job	count(*)
TA	2
Prof	2

```
SELECT job, count(*)  
FROM payroll  
WHERE salary > 55000  
GROUP BY job;
```

job	count(*)
TA	1
Prof	2

```
SELECT job, count(*)  
FROM payroll  
WHERE salary > 75000  
GROUP BY job;
```

TA group  
no longer  
exists

job	count(*)
Prof	2

## payroll

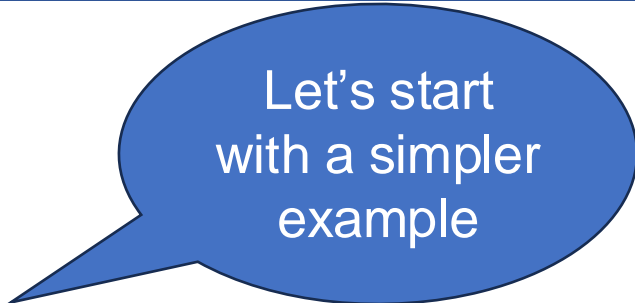
user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Can never have count(\*)=0  
If we want them: outer joins!



# Coping with Empty Groups

How many cars does each person drive?



Let's start  
with a simpler  
example

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

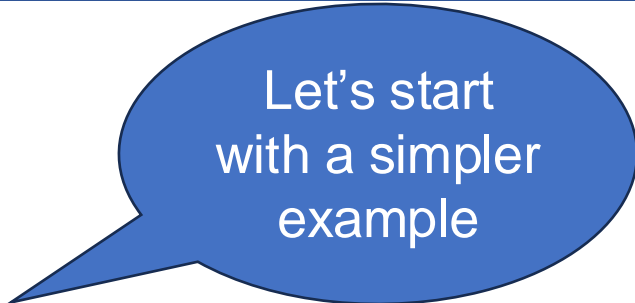
**Regist**

user_id	Car
123	Charger
567	Civic
567	Pinto

# Coping with Empty Groups

How many cars does each person drive?

```
SELECT P.name, count(*)
FROM payroll P, Regist R
WHERE P.user_id = R.user_id
GROUP BY P.user_id;
```



Let's start  
with a simpler  
example

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

**Regist**

user_id	Car
123	Charger
567	Civic
567	Pinto

# Coping with Empty Groups

How many cars does each person drive?

```
SELECT P.name, count(*)
FROM payroll P, Regist R
WHERE P.user_id = R.user_id
GROUP BY P.user_id;
```

Let's start with a simpler example

We want this



name	count
Jack	1
Magda	2

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

user_id	Car
123	Charger
567	Civic
567	Pinto

# Coping with Empty Groups

How many cars does each person drive?

```
SELECT P.name, count(*)
FROM payroll P, Regist R
WHERE P.user_id = R.user_id
GROUP BY P.user_id;
```

Incorrect!

Why??

We want this



name	count
Jack	1
Magda	2

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

user_id	Car
123	Charger
567	Civic
567	Pinto

# Coping with Empty Groups

How many cars does each person drive?

```
SELECT P.name, count(*)
FROM payroll P, Regist R
WHERE P.user_id = R.user_id
GROUP BY P.user_id;
```

Incorrect!

Why??

P.name must occur in GROUP BY

We want this



name	count
Jack	1
Magda	2

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

user_id	Car
123	Charger
567	Civic
567	Pinto

# Coping with Empty Groups

How many cars does each person drive?

```
SELECT P.name, count(*)  
FROM payroll P, Regist R  
WHERE P.user_id = R.user_id  
GROUP BY P.name, P.user_id;
```

Now it's correct

We want this



name	count
Jack	1
Magda	2

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

user_id	Car
123	Charger
567	Civic
567	Pinto

# Coping with Empty Groups

How many cars does each person drive?

```
SELECT P.name, count(*)
FROM payroll P, Regist R
WHERE P.user_id = R.user_id
GROUP BY P.name, P.user_id;
```

Steps 1,2:

P.user_id	P.name	P.job	P.salary	R.user_id	R.Car
123	Jack	TA	50000	123	Charger
567	Magda	Prof	90000	567	Civic
567	Magda	Prof	90000	567	Pinto

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

user_id	Car
123	Charger
567	Civic
567	Pinto

# Coping with Empty Groups

How many cars does each person drive?

```
SELECT P.name, count(*)
FROM payroll P, Regist R
WHERE P.user_id = R.user_id
GROUP BY P.name, P.user_id;
```

Steps 1,2:

P.user_id	P.name	P.job	P.salary	R.user_id	R.Car
123	Jack	TA	50000	123	Charger
567	Magda	Prof	90000	567	Civic
567	Magda	Prof	90000	567	Pinto



name	count
Jack	1
Magda	2

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

user_id	Car
123	Charger
567	Civic
567	Pinto



# Coping with Empty Groups

How many cars does each person drive?

```
SELECT P.name, count(*)
FROM payroll P, Regist R
WHERE P.user_id = R.user_id
GROUP BY P.name, P.user_id;
```

To also include Allison, Dan,  
we will use outer joins

Steps 1,2:

P.user_id	P.name	P.job	P.salary	R.user_id	R.Car
123	Jack	TA	50000	123	Charger
567	Magda	Prof	90000	567	Civic
567	Magda	Prof	90000	567	Pinto

name	count
Jack	1
Magda	2

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

user_id	Car
123	Charger
567	Civic
567	Pinto

# Coping with Empty Groups

How many cars does each person drive?

To also include Allison, Dan,  
we will use outer joins

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

**Regist**

user_id	Car
123	Charger
567	Civic
567	Pinto

# Coping with Empty Groups

How many cars does each person drive?

```
SELECT P.name, count(*)  
FROM payroll P LEFT OUTER JOIN Regist R ON P.user_id = R.user_id  
GROUP BY P.name, P.user_id;
```

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

**Regist**

user_id	Car
123	Charger
567	Civic
567	Pinto

# Coping with Empty Groups

How many cars does each person drive?

```
SELECT P.name, count(*)  
FROM payroll P LEFT OUTER JOIN Regist R ON P.user_id = R.user_id  
GROUP BY P.name, P.user_id;
```

Step 1

P.user_id	P.name	P.job	P.salary	R.user_id	R.Car
123	Jack	TA	50000	123	Charger
345	Allison	TA	60000	NULL	NULL
567	Magda	Prof	90000	567	Civic
567	Magda	Prof	90000	567	Pinto
789	Dan	Prof	100000	NULL	NULL

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

user_id	Car
123	Charger
567	Civic
567	Pinto

# Coping with Empty Groups

How many cars does each person drive?

```
SELECT P.name, count(*)  
FROM payroll P LEFT OUTER JOIN Regist R ON P.user_id = R.user_id  
GROUP BY P.name, P.user_id;
```

Steps 1,2:

P.user_id	P.name	P.job	P.salary	R.user_id	R.Car
123	Jack	TA	50000	123	Charger
345	Allison	TA	60000	NULL	NULL
567	Magda	Prof	90000	567	Civic
567	Magda	Prof	90000	567	Pinto
789	Dan	Prof	100000	NULL	NULL

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

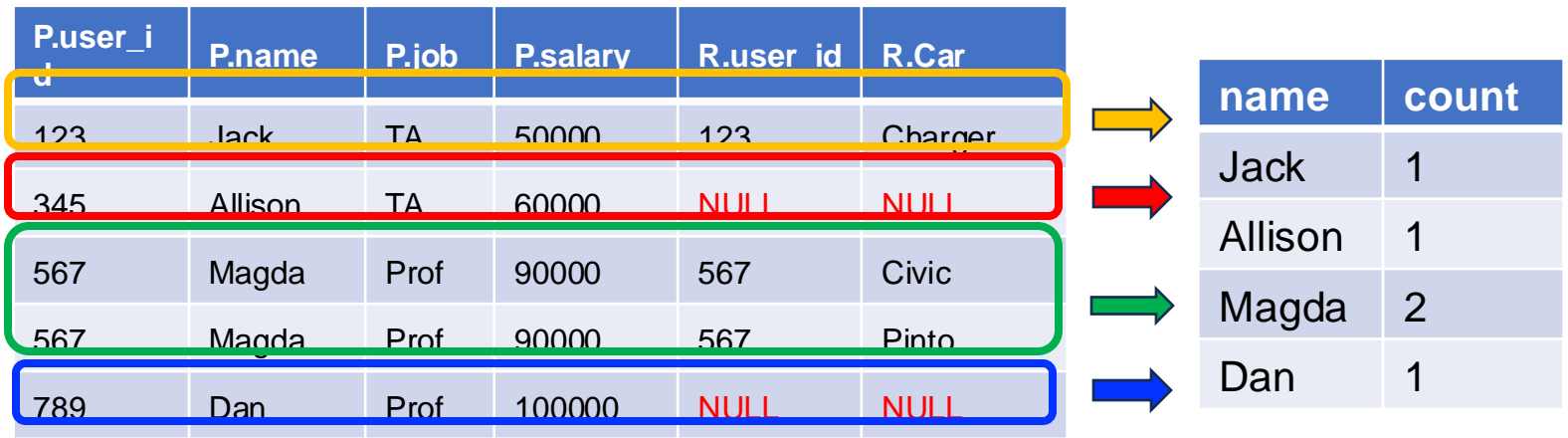
user_id	Car
123	Charger
567	Civic
567	Pinto

# Coping with Empty Groups

How many cars does each person drive?

```
SELECT P.name, count(*)  
FROM payroll P LEFT OUTER JOIN Regist R ON P.user_id = R.user_id  
GROUP BY P.name, P.user_id;
```

Steps 1,2:



**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

**Regist**

user_id	Car
123	Charger
567	Civic
567	Pinto

# Coping with Empty Groups

How many cars does each person drive?

```
SELECT P.name, count(*)
FROM payroll P LEFT OUTER JOIN Regist R ON P.user_id = R.user_id
GROUP BY P.name, P.user_id;
```

Steps 1,2:

P.user_id	P.name	P.job	P.salary	R.user_id	R.Car
123	Jack	TA	50000	123	Charger
345	Allison	TA	60000	NULL	NULL
567	Magda	Prof	90000	567	Civic
567	Magda	Prof	90000	567	Pinto
789	Dan	Prof	100000	NULL	NULL

name	count
Jack	1
Allison	1
Magda	2
Dan	1

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

user_id	Car
123	Charger
567	Civic
567	Pinto

Should be 0!  
How to fix?

# Coping with Empty Groups

How many cars does each person drive?

count ignores NULLs

```
SELECT P.name, count(R.user_id)
FROM payroll P LEFT OUTER JOIN Regist R ON P.user_id = R.user_id
GROUP BY P.name, P.user_id;
```

Steps 1,2:

P.user_id	P.name	P.job	P.salary	R.user_id	R.Car
123	Jack	TA	50000	123	Charger
345	Allison	TA	60000	NULL	NULL
567	Magda	Prof	90000	567	Civic
567	Magda	Prof	90000	567	Pinto
789	Dan	Prof	100000	NULL	NULL



name	count
Jack	1
Allison	0
Magda	2
Dan	0

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

user_id	Car
123	Charger
567	Civic
567	Pinto

Now it's correct



# Coping with Empty Groups

For each job, how many people earn more than 75000?

## payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# Coping with Empty Groups

For each job, how many people earn more than 75000?

```
SELECT job, count(*)  
FROM payroll  
WHERE salary > 75000  
GROUP BY job;
```

job	count(*)
Prof	2

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# Coping with Empty Groups

For each job, how many people earn more than 75000?

```
SELECT job, count(*)  
FROM payroll  
WHERE salary > 75000  
GROUP BY job;
```

job	count(*)
Prof	2

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

To include users where  $\text{count}(*)=0$ , we will use a self-outer-join

# Coping with Empty Groups

For each job, how many people earn more than 75000?

```
SELECT P1.job, count(
)
FROM payroll P1 LEFT OUTER JOIN payroll P2
ON P1.job = P2.job and P2.salary > 75000
GROUP BY P1.job;
```

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

To include users where `count(*)=0`, we will use a self-outer-join

# Coping with Empty Groups

For each job, how many people earn more than 75000?

What goes here?

```
SELECT P1.job, count(
)
FROM payroll P1 LEFT OUTER JOIN payroll P2
  ON P1.job = P2.job and P2.salary > 75000
GROUP BY P1.job;
```

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# Coping with Empty Groups

For each job, how many people earn more than 75000?

```
SELECT P1.job, count(
)
FROM payroll P1 LEFT OUTER JOIN payroll P2
ON P1.job = P2.job and P2.salary > 75000
GROUP BY P1.job;
```

Left Outer Join

P1.user_id	P1.name	P1.job	P1.salary	P2.user_id	P2.name	P2.job	P2.salary
123	Jack	TA	50000	NULL	NULL	NULL	NULL
345	Allison	TA	60000	NULL	NULL	NULL	NULL
567	Magda	Prof	90000	567	Magda	Prof	90000
789	Dan	Prof	100000	567	Magda	Prof	90000
567	Magda	Prof	90000	789	Dan	Prof	100000
789	Dan	Prof	100000	789	Dan	Prof	100000

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

We want to include all jobs,  
even when the count is 0.  
Need an outer join with the jobs

# Coping with Empty Groups

For each job, how many people earn more than 75000?

```
SELECT P1.job, count(
)
FROM payroll P1 LEFT OUTER JOIN payroll P2
ON P1.job = P2.job and P2.salary > 75000
GROUP BY P1.job;
```

Group by P1.job

P1.user_id	P1.name	P1.job	P1.salary	P2.user_id	P2.name	P2.job	P2.salary
123	Jack	TA	50000	NULL	NULL	NULL	NULL
345	Allison	TA	60000	NULL	NULL	NULL	NULL
567	Magda	Prof	90000	567	Magda	Prof	90000
789	Dan	Prof	100000	567	Magda	Prof	90000
567	Magda	Prof	90000	789	Dan	Prof	100000
789	Dan	Prof	100000	789	Dan	Prof	100000

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

We want to include all jobs, even when the count is 0. Need an outer join with the jobs

# Coping with Empty Groups

For each job, how many people earn more than 75000

What do we write here?

```
SELECT P1.job, count(
FROM payroll P1 LEFT OUTER JOIN payroll P2
ON P1.job = P2.job and P2.salary > 75000
GROUP BY P1.job;
```

Want this:

job	...
TA	0
Prof	2

P1.user_id	P1.name	P1.job	P1.salary	P2.user_id	P2.name	P2.job	P2.salary
123	Jack	TA	50000	NULL	NULL	NULL	NULL
345	Allison	TA	60000	NULL	NULL	NULL	NULL
567	Magda	Prof	90000	567	Magda	Prof	90000
789	Dan	Prof	100000	567	Magda	Prof	90000
567	Magda	Prof	90000	789	Dan	Prof	100000
789	Dan	Prof	100000	789	Dan	Prof	100000

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

We want to include all jobs,  
even when the count is 0.  
Need an outer join with the jobs



# Coping with Empty Groups

For each job, how many people earn more than 75000?

```
SELECT P1.job, count(DISTINCT P2.user_id)
FROM payroll P1 LEFT OUTER JOIN payroll P2
  ON P1.job = P2.job and P2.salary > 75000
GROUP BY P1.job;
```

count this

P1.user_id	P1.name	P1.job	P1.salary	P2.user_id	P2.name	P2.job	P2.salary
123	Jack	TA	50000	NULL	NULL	NULL	NULL
345	Allison	TA	60000	NULL	NULL	NULL	NULL
567	Magda	Prof	90000	567	Magda	Prof	90000
789	Dan	Prof	100000	567	Magda	Prof	90000
567	Magda	Prof	90000	789	Dan	Prof	100000
789	Dan	Prof	100000	789	Dan	Prof	100000

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

We want to include all jobs,  
even when the count is 0.  
Need an outer join with the jobs

# Coping with Empty Groups

For each job, how many people earn more than 75000?

```
SELECT P1.job, count(DISTINCT P2.user_id)
FROM payroll P1 LEFT OUTER JOIN payroll P2
  ON P1.job = P2.job and P2.salary > 75000
GROUP BY P1.job;
```

job	count(...)
TA	0
Prof	2

P1.user_id	P1.name	P1.job	P1.salary	P2.user_id	P2.name	P2.job	P2.salary
123	Jack	TA	50000	NULL	NULL	NULL	NULL
345	Allison	TA	60000	NULL	NULL	NULL	NULL
567	Magda	Prof	90000	567	Magda	Prof	90000
789	Dan	Prof	100000	567	Magda	Prof	90000
567	Magda	Prof	90000	789	Dan	Prof	100000
789	Dan	Prof	100000	789	Dan	Prof	100000



**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

We want to include all jobs, even when the count is 0. Need an outer join with the jobs

Coping with empty groups requires some creativity

- Use Left-outer-join
- Sometimes, you need a self-left-outer-join

# The HAVING Clause

# The HAVING Clause

- **WHERE:**

- Applies a predicate to a single tuple\*
- Cannot use any aggregate operation

- **HAVING:**

- Applies a predicate to an entire group
- May use aggregate operations
- Can only check attributes occurring in GROUP-BY

\* Actually, to one tuple from each relation in the FROM clause

# The HAVING Clause

Find the total quantity of products that were sold  $\geq 2$  times.

## sales

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March

# The HAVING Clause

Find the total quantity of products that were sold  $\geq 2$  times.

```
SELECT product, sum(quant)
FROM sales
GROUP BY product
HAVING count(*)  $\geq$  2;
```

## sales

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March

# The HAVING Clause

Find the total quantity of products that were sold  $\geq 2$  times.

```
SELECT product, sum(quant)
FROM sales
GROUP BY product
HAVING count(*)  $\geq$  2;
```

## sales

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March



# The HAVING Clause

Find the total quantity of products that were sold  $\geq 2$  times.

```
SELECT product, sum(quant)
FROM sales
GROUP BY product
HAVING count(*)  $\geq$  2;
```

## sales

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March

count(\*)=3

count(\*)=2

count(\*)=1 NOT included

# The HAVING Clause

Find the total quantity of products that were sold  $\geq 2$  times.

```
SELECT product, sum(quant)
FROM sales
GROUP BY product
HAVING count(*)  $\geq$  2;
```

## sales

product	price	quant	month
Bagel	3	20	Jan
Bagel	5	10	Jan
Bagel	1.50	20	March
Banana	0.5	50	Feb
Banana	5	10	Feb
Apple	4	10	March

count(\*)=3

count(\*)=2

count(\*)=1 NOT included



product	sum
Bagel	50
Banana	60

# SQL Query Summary

```
SELECT A  
FROM R1, ..., Rn  
WHERE C1  
GROUP BY a1, ..., ak  
HAVING C2  
ORDER BY T
```

A = any attributes from  $a_1, \dots, a_k$  and/or any aggregates

C1 = any condition on the attributes in  $R_1, \dots, R_n$

C2 = any condition on  $a_1, \dots, a_k$  and/or any aggregates

T = any attributes from  $a_1, \dots, a_k$  and/or any aggregates

# Discussion: WHERE v.s. HAVING

## ■ WHERE:

- Applies to single tuple from each table
- May decrease size of groups, even make them empty
- Cannot use aggregates (count(\*)=5, sum(...) > 10)

## ■ HAVING:

- Applies to entire group: keep it or drop it
- May use aggregates (count(\*)=5, sum(...) > 10)
- May only use attributes in GROUP-BY

# The Witness

# The Witness

- SQL provides the aggregate operators min, max
- SQL does not have argmin or argmax
- Often we want to find the record that achieves that minimum or maximum: we call it **The Witness**
- One way to compute it is using the HAVING clause
- A simpler way discussed later

# The Witnessing Problem

Find the person with highest salary for each job

## payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# The Witnessing Problem

Find the person with highest salary for each job

Desired answer:

job	name	salary
TA	Allison	60000
Prof	Dan	100000

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000



# The Witnessing Problem

Find the person with highest salary for each job

```
SELECT job, MAX(salary)
FROM payroll
GROUP BY job
```

job	salary
TA	60000
Prof	100000

Finding max is easy.

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# The Witnessing Problem

Find the person with highest salary for each job

```
SELECT job, MAX(salary)
FROM payroll
GROUP BY job
```

job	salary
TA	60000
Prof	100000

Finding max is easy.

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

But we want argmax.  
How do we find  
the witness?

# The Witnessing Problem

Find the person with highest salary for each job

```
SELECT job, name, MAX(salary)
FROM payroll
GROUP BY job
```

Does this work?

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# The Witnessing Problem

Find the person with highest salary for each job

```
SELECT job, name, MAX(salary)
FROM payroll
GROUP BY job
```

Does this work?

**WRONG!**  
name not in GROUP BY

payroll

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Sqlite does not return an error, but returns junk outputs. Don't use this.

# The Witnessing Problem

Find the person with highest salary for each job

Plan:

1. Compute the  $\max(\text{salary})$  for each job
2. Join back with payroll on job
3. Return the users where  $\text{salary} = \max(\text{salary})$

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# The Witnessing Problem

Find the person with highest salary for each job

Plan:

1. Compute the  $\max(\text{salary})$  for each job
2. Join back with payroll on job
3. Return the users where  $\text{salary} = \max(\text{salary})$

We first join

Goes in HAVING

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# The Witnessing Problem

Find the person with highest salary for each job

```
SELECT P1.job, MAX(P1.salary)
FROM payroll AS P1

GROUP BY P1.job
```

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# The Witnessing Problem

Find the person with highest salary for each job

```
SELECT P1.job, MAX(P1.salary)  
FROM payroll AS P1  
  
GROUP BY P1.job
```

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000



# The Witnessing Problem

Find the person with highest salary for each job

```
SELECT P1.job  
FROM payroll AS P1  
  
GROUP BY P1.job
```

**payroll**

<b>user_id</b>	<b>name</b>	<b>job</b>	<b>salary</b>
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# The Witnessing Problem

Find the person with highest salary for each job

```
SELECT P1.job, P2.name, P2.salary
FROM payroll AS P1, payroll AS P2
WHERE P1.job = P2.job
GROUP BY P1.job
```

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# The Witnessing Problem

Find the person with highest salary for each job

```
SELECT P1.job, P2.name, P2.salary
FROM payroll AS P1, payroll AS P2
WHERE P1.job = P2.job
GROUP BY P1.job
```

Incorrect!

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# The Witnessing Problem

Find the person with highest salary for each job

```
SELECT P1.job, P2.name, P2.salary
FROM payroll AS P1, payroll AS P2
WHERE P1.job = P2.job
GROUP BY P1.job, P2.name, P2.salary
```

Correct; but not done!

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# The Witnessing Problem

Find the person with highest salary for each job

```
SELECT P1.job, P2.name, P2.salary
FROM payroll AS P1, payroll AS P2
WHERE P1.job = P2.job
GROUP BY P1.job, P2.name, P2.salary
```

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Which P2 should we return for each job?

# The Witnessing Problem

Find the person with highest salary for each job

```
SELECT P1.job, P2.name, P2.salary
FROM payroll AS P1, payroll AS P2
WHERE P1.job = P2.job
GROUP BY P1.job, P2.name, P2.salary
HAVING P2.salary = MAX(P1.salary)
```

**payroll**

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# The Witnessing Problem

Find the person with highest salary for each job

```
SELECT P1.job, P2.name, P2.salary
FROM payroll AS P1, payroll AS P2
WHERE P1.job = P2.job
GROUP BY P1.job, P2.name, P2.salary
HAVING MAX(P1.salary) = P2.salary;
```

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# The Witnessing Problem

Find the person with highest salary for each job

```
SELECT P1.job, P2.name, P2.salary
FROM payroll AS P1, payroll AS P2
WHERE P1.job = P2.job
GROUP BY P1.job, P2.name, P2.salary
HAVING MAX(P1.salary) = P2.salary;
```

payroll join with payroll

P1				P2			
user_id	name	job	salary	user_id	name	job	salary
123	Jack	TA	50000	123	Jack	TA	50000
345	Allison	TA	60000	123	Jack	TA	50000
123	Jack	TA	50000	345	Allison	TA	60000
345	Allison	TA	60000	345	Allison	TA	60000
567	Magda	Prof	90000	567	Magda	Prof	90000
789	Dan	Prof	100000	567	Magda	Prof	90000
567	Magda	Prof	90000	789	Dan	Prof	100000
789	Dan	Prof	100000	789	Dan	Prof	100000

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000



# The Witnessing Problem

Find the person with highest salary for each job

```
SELECT P1.job, P2.name, P2.salary
FROM payroll AS P1, payroll AS P2
WHERE P1.job = P2.job
GROUP BY P1.job, P2.name, P2.salary
HAVING MAX(P1.salary) = P2.salary;
```

Group by

P1				P2			
user_id	name	job	salary	user_id	name	job	salary
123	Jack	TA	50000	123	Jack	TA	50000
345	Allison	TA	60000	123	Jack	TA	50000
123	Jack	TA	50000	345	Allison	TA	60000
345	Allison	TA	60000	345	Allison	TA	60000
567	Magda	Prof	90000	567	Magda	Prof	90000
789	Dan	Prof	100000	567	Magda	Prof	90000
567	Magda	Prof	90000	789	Dan	Prof	100000
789	Dan	Prof	100000	789	Dan	Prof	100000

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# The Witnessing Problem

Find the person with highest salary for each job

```
SELECT P1.job, P2.name, P2.salary
FROM payroll AS P1, payroll AS P2
WHERE P1.job = P2.job
GROUP BY P1.job, P2.name, P2.salary
HAVING MAX(P1.salary) = P2.salary;
```

Compute max(P1.salary)

P1				P2			
user_id	name	job	salary	user_id	name	job	salary
123	Jack	TA	50000	123	Jack	TA	50000
345	Allison	TA	60000	123	Jack	TA	50000
123	Jack	TA	50000	345	Allison	TA	60000
345	Allison	TA	60000	345	Allison	TA	60000
567	Magda	Prof	90000	567	Magda	Prof	90000
789	Dan	Prof	100000	567	Magda	Prof	90000
567	Magda	Prof	90000	789	Dan	Prof	100000
789	Dan	Prof	100000	789	Dan	Prof	100000

max(salary)=60000

max(salary)=60000

max(salary)=100000

max(salary)=100000

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# The Witnessing Problem

Find the person with highest salary for each job

```
SELECT P1.job, P2.name, P2.salary
FROM payroll AS P1, payroll AS P2
WHERE P1.job = P2.job
GROUP BY P1.job, P2.name, P2.salary
HAVING MAX(P1.salary) = P2.salary;
```



P1				P2			
user_id	name	job	salary	user_id	name	job	salary
123	Jack	TA	50000	123	Jack	TA	50000
345	Allison	TA	60000	123	Jack	TA	50000
123	Jack	TA	50000	345	Allison	TA	60000
345	Allison	TA	60000	345	Allison	TA	60000
567	Magda	Prof	90000	567	Magda	Prof	90000
789	Dan	Prof	100000	567	Magda	Prof	90000
567	Magda	Prof	90000	789	Dan	Prof	100000
789	Dan	Prof	100000	789	Dan	Prof	100000

max(salary)=60000

max(salary)=60000

max(salary)=100000

max(salary)=100000

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

# The Witnessing Problem

Find the person with highest salary for each job

```
SELECT P1.job, P2.name, P2.salary
FROM payroll AS P1, payroll AS P2
WHERE P1.job = P2.job
GROUP BY P1.job, P2.name, P2.salary
HAVING MAX(P1.salary) = P2.salary;
```

P1				P2			
user_id	name	job	salary	user_id	name	job	salary
123	Jack	TA	50000	123	Jack	TA	50000
345	Allison	TA	60000	123	Jack	TA	50000
123	Jack	TA	50000	345	Allison	TA	60000
345	Allison	TA	60000	345	Allison	TA	60000
567	Magda	Prof	90000	567	Magda	Prof	90000
789	Dan	Prof	100000	567	Magda	Prof	90000
567	Magda	Prof	90000	789	Dan	Prof	100000
789	Dan	Prof	100000	789	Dan	Prof	100000

max(salary)=60000

max(salary)=60000

max(salary)=100000

max(salary)=100000

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000



P1.job	P2.name	P2.salary
TA	Allison	60000
Prof	Dan	100000

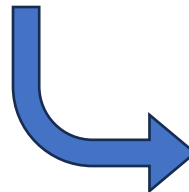
# The Witnessing Problem

Find the person with highest salary for each job

```
SELECT P1.job, P2.name, P2.salary
FROM payroll AS P1, payroll AS P2
WHERE P1.job = P2.job
GROUP BY P1.job, P2.name, P2.salary
HAVING MAX(P1.salary) = P2.salary;
```

Final output has the witnesses

user_id	name	job	salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000



P1.job	P2.name	P2.salary
TA	Allison	60000
Prof	Dan	100000

# Summary

Group-by can be subtle!

- Empty groups
- Having clause
- Finding the witness