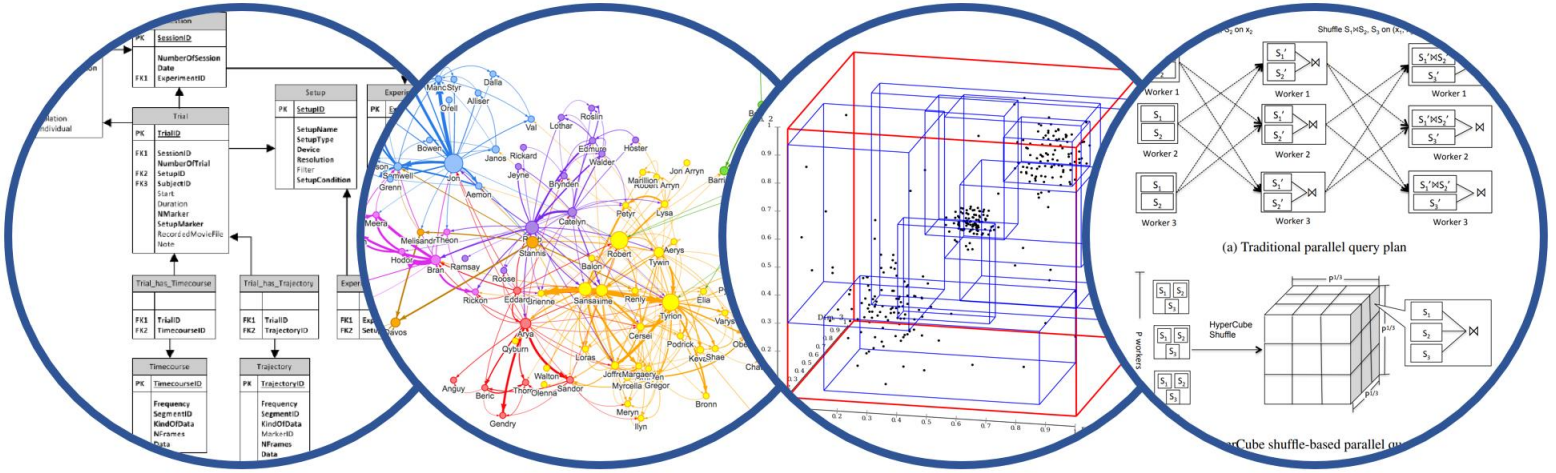


CSE 344: Intro to Data Management

Aggregates

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

Announcements

- Homework 2
  - Posted
  - Due on Wednesday
  - Sqlite

- Homework 3: coming up soon (SQL Azure)

Today's lecture is more challenging!
Please study the slides carefully at home
Will review the "witness problem" on Monday

Aggregates

```
SELECT count(*) as C
FROM Payroll
WHERE Job = 'TA';
```

May use alias

C
2

Payroll

UserID	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

We may compute several aggreges

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Today: 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

UserID	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

UserID	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

UserID	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

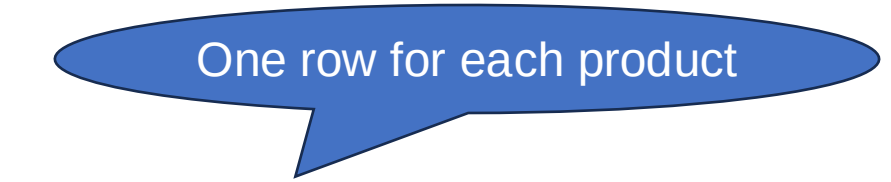
Group By Basics

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



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

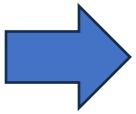
Group By Basics

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

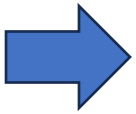
Group By Basics

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

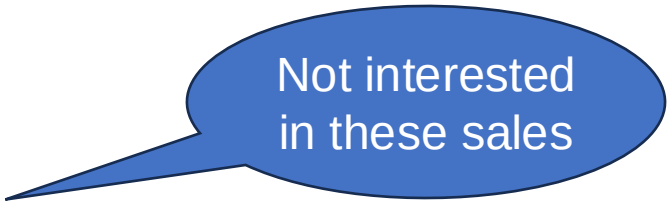
Group By Basics

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~~	~~1.50~~	~~20~~	~~March~~
~~Banana~~	~~0.5~~	~~50~~	~~Feb~~
Banana	5	10	Feb
Apple	4	10	March



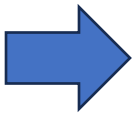
Group By Basics

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~~	~~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	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~~	~~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	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;
```

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

Step 1

```
SELECT *
FROM Sales
WHERE Price < 4.5;
```

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;
```

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

Step 1

```
SELECT *
FROM Sales
WHERE Price < 4.5;
```

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

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

Step 1

```
SELECT *
FROM Sales
WHERE Price < 4.5;
```

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

Step 3

Month	Quant
Jan	20
Feb	50
March	30

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 count(*) ≥ 1
- Sometimes we want answers with count(…)=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

UserID	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

UserID	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

UserID	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

UserID	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;
```

Job	Count(*)
Prof	2

Payroll

UserID	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

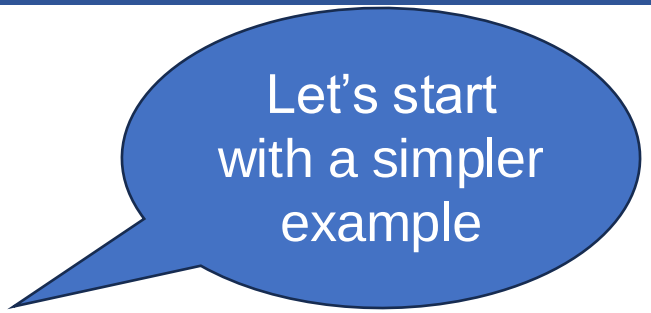
Payroll

UserID	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?



Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

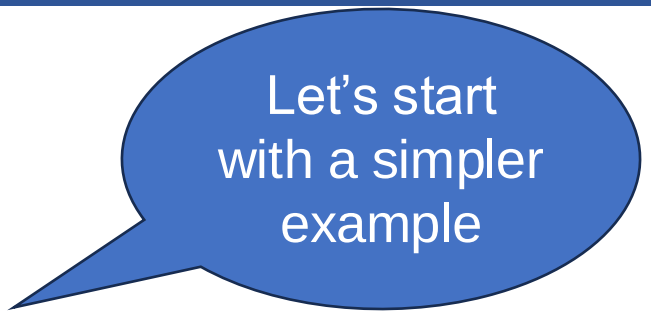
Regist

UserID	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.UserID = R.UserID
GROUP BY P.UserID;
```



Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	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.UserID = R.UserID
GROUP BY P.UserID;
```

Let's start with a simpler example

We want this

Name	count
Jack	1
Magda	2

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	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.UserID = R.UserID
GROUP BY P.UserID;
```

Why??

We want this

Name	count
Jack	1
Magda	2

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	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.UserID = R.UserID
GROUP BY P.UserID;
```

Incorrect!

Why??

P.Name must occur in GROUP BY

We want this

Name	count
Jack	1
Magda	2

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	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.UserID = R.UserID
GROUP BY P.Name, P.UserID;
```

Now it's correct

We want this

Name	count
Jack	1
Magda	2

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	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.UserID = R.UserID
GROUP BY P.Name, P.UserID;
```

Steps 1,2:

P.UserID	P.Name	P.Job	P.Salary	R.UserID	R.Car
123	Jack	TA	50000	123	Charger
567	Magda	Prof	90000	567	Civic
567	Magda	Prof	90000	567	Pinto

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	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.UserID = R.UserID
GROUP BY P.Name, P.UserID;
```

Steps 1,2:

P.UserID	P.Name	P.Job	P.Salary	R.UserID	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

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	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.UserID = R.UserID
GROUP BY P.Name, P.UserID;
```

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

Steps 1,2:

P.UserID	P.Name	P.Job	P.Salary	R.UserID	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

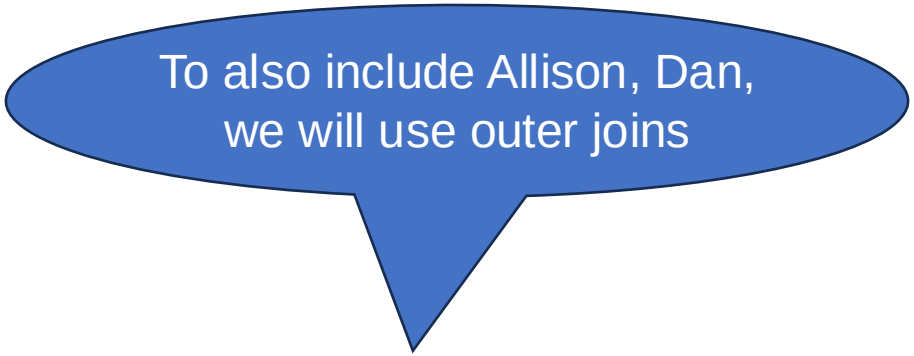
UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Coping with Empty Groups

How many cars does each person drive?



Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	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.UserID = R.UserID
GROUP BY P.Name, P.UserID;
```

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	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.UserID = R.UserID
GROUP BY P.Name, P.UserID;
```

Step 1

P.UserID	P.Name	P.Job	P.Salary	R.UserID	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

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	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.UserID = R.UserID
GROUP BY P.Name, P.UserID;
```

Steps 1,2:

P.UserID	P.Name	P.Job	P.Salary	R.UserID	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

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	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.UserID = R.UserID
GROUP BY P.Name, P.UserID;
```

Steps 1,2:

P.UserID	P.Name	P.Job	P.Salary	R.UserID	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

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	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.UserID = R.UserID
GROUP BY P.Name, P.UserID;
```

Steps 1,2:

P.UserID	P.Name	P.Job	P.Salary	R.UserID	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

Should be 0!
How to fix?

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Coping with Empty Groups

How many cars does each person drive?

Count ignores NULLs

```
SELECT P.Name, count(R.UserID)
FROM Payroll P LEFT OUTER JOIN Regist R ON P.UserID = R.UserID
GROUP BY P.Name, P.UserID;
```

Steps 1,2:

P.UserID	P.Name	P.Job	P.Salary	R.UserID	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

Now it's correct

Payroll

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

Regist

UserID	Car
123	Charger
567	Civic
567	Pinto

Coping with Empty Groups

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

Payroll

UserID	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

UserID	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

UserID	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?

```
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

UserID	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?

```
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

UserID	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;
```

Payroll

UserID	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.UserID	P1.Name	P1.Job	P1.Salary	P2.UserID	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

UserID	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.UserID	P1.Name	P1.Job	P1.Salary	P2.UserID	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

UserID	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.UserID	P1.Name	P1.Job	P1.Salary	P2.UserID	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

UserID	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.UserID)
FROM Payroll P1 LEFT OUTER JOIN Payroll P2
  ON P1.Job = P2.Job and P2.Salary > 75000
GROUP BY P1.Job;
```

Count this

P1.UserID	P1.Name	P1.Job	P1.Salary	P2.UserID	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

UserID	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.UserID)
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.UserID	P1.Name	P1.Job	P1.Salary	P2.UserID	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

UserID	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

Discussion

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 ≥ 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 ≥ 2 times.

```
SELECT Product, sum(Quant)
FROM Sales
GROUP BY Product
HAVING count(*) >= 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 ≥ 2 times.

```
SELECT Product, sum(Quant)
FROM Sales
GROUP BY Product
HAVING count(*) >= 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 ≥ 2 times.

```
SELECT Product, sum(Quant)
FROM Sales
GROUP BY Product
HAVING count(*) >= 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 ≥ 2 times.

```
SELECT Product, sum(Quant)
FROM Sales
GROUP BY Product
HAVING count(*) >= 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, …, a_k and/or any aggregates

C1 = any condition on the attributes in R_1, …, R_n

C2 = any condition on a_1, …, a_k and/or any aggregates

T = any attributes from a_1, …, 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 on Monday

The Witnessing Problem

Find the person with highest salary for each job

Payroll

UserID	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

UserID	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

UserID	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

UserID	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

UserID	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

UserID	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(Salary) for each Job
2. Join back with Payroll on Job
3. Return the users where Salary = max(Salary)

Payroll

UserID	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(Salary) for each Job
2. Join back with Payroll on Job
3. Return the users where Salary = max(Salary)

We first join

Goes in HAVING

Payroll

UserID	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

UserID	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

UserID	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

UserID	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
```

Payroll

UserID	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
```

Payroll

UserID	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

UserID	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
```

Which P2 should we return for each Job?

Payroll

UserID	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 P2.Salary = MAX(P1.Salary)
```

Payroll

UserID	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;
```

UserID	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

UserID	Name	Job	Salary	UserID	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

UserID	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

UserID	Name	Job	Salary	UserID	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

UserID	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

UserID	Name	Job	Salary	UserID	Name	Job	Salary	
123	Jack	TA	50000	123	Jack	TA	50000	max(salary)=60000
345	Allison	TA	60000	123	Jack	TA	50000	
123	Jack	TA	50000	345	Allison	TA	60000	max(salary)=60000
345	Allison	TA	60000	345	Allison	TA	60000	
567	Magda	Prof	90000	567	Magda	Prof	90000	max(salary)=100000
789	Dan	Prof	100000	567	Magda	Prof	90000	
567	Magda	Prof	90000	789	Dan	Prof	100000	max(salary)=100000
789	Dan	Prof	100000	789	Dan	Prof	100000	

UserID	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;
```

Check HAVING

P1 | P2

UserID	Name	Job	Salary	UserID	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

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

The Witnessing Problem

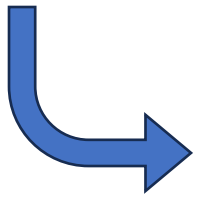
Find the person with highest salary for each job

```
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

UserID	Name	Job	Salary	UserID	Name	Job	Salary	
123	Jack	TA	50000	123	Jack	TA	50000	max(salary)=60000
345	Allison	TA	60000	123	Jack	TA	50000	
123	Jack	TA	50000	345	Allison	TA	60000	max(salary)=60000
345	Allison	TA	60000	345	Allison	TA	60000	
567	Magda	Prof	90000	567	Magda	Prof	90000	max(salary)=100000
789	Dan	Prof	100000	567	Magda	Prof	90000	
567	Magda	Prof	90000	789	Dan	Prof	100000	max(salary)=100000
789	Dan	Prof	100000	789	Dan	Prof	100000	

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000



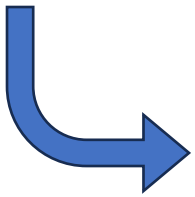
P1.Job	P2.Name	P2.Salary
TA	Allison	60000
Prof	Dan	100000

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;
```

UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000



Final output has the witnesses

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