



# Announcements

- HW 1 due Friday @11pm on Gradescope
- HW 2 releases Friday! Due Monday 1/22 @11pm

# Self-Joins

Find all people who drive a Civic and Pinto

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

UserID	Car
123	Charger
567	Civic
567	Pinto

```
SELECT P.Name
FROM Payroll AS P, Regist AS R
WHERE P.UserID = R.UserID AND
      R.Car = 'Civic';
```

# Self-Joins

Find all people who drive a Civic and Pinto

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

UserID	Car
123	Charger
567	Civic
567	Pinto

```
SELECT P.Name, R.Car
FROM Payroll AS P, Regist AS R
WHERE P.UserID = R.UserID AND
      R.Car = 'Civic' AND
      R.Car = 'Pinto' ;
```

Will this work?

# Self-Joins

Find all people who drive a Civic and Pinto

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

UserID	Car
123	Charger
567	Civic
567	Pinto

```
SELECT P.Name, R.Car
FROM Payroll AS P, Regist AS R
WHERE P.UserID = R.UserID AND
      R.Car = 'Civic' AND
      R.Car = 'Pinto';
```

Will this work?

Nope, empty set is  
returned

# Self-Joins

Find all people who drive a Civic and Pinto

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

UserID	Car
123	Charger
567	Civic
567	Pinto

```
SELECT P.Name, R1.Car, R2.Car
FROM Payroll AS P, Regist AS R1, Regist AS R2
WHERE P.UserID = R1.UserID AND
      P.UserID = R2.UserID AND
      R1.Car = 'Civic' AND
      R2.Car = 'Pinto';
```

# Self-Joins

Find all people who drive a Civic and Pinto

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

UserID	Car
123	Charger
567	Civic
567	Pinto

All pairs of cars a person can drive

```
SELECT P.Name, R1.Car, R2.Car
FROM Payroll AS P, Regist AS R1, Regist AS R2
WHERE P.UserID = R1.UserID AND
      P.UserID = R2.UserID AND
      R1.Car = 'Civic' AND
      R2.Car = 'Pinto';
```

# Self-Joins

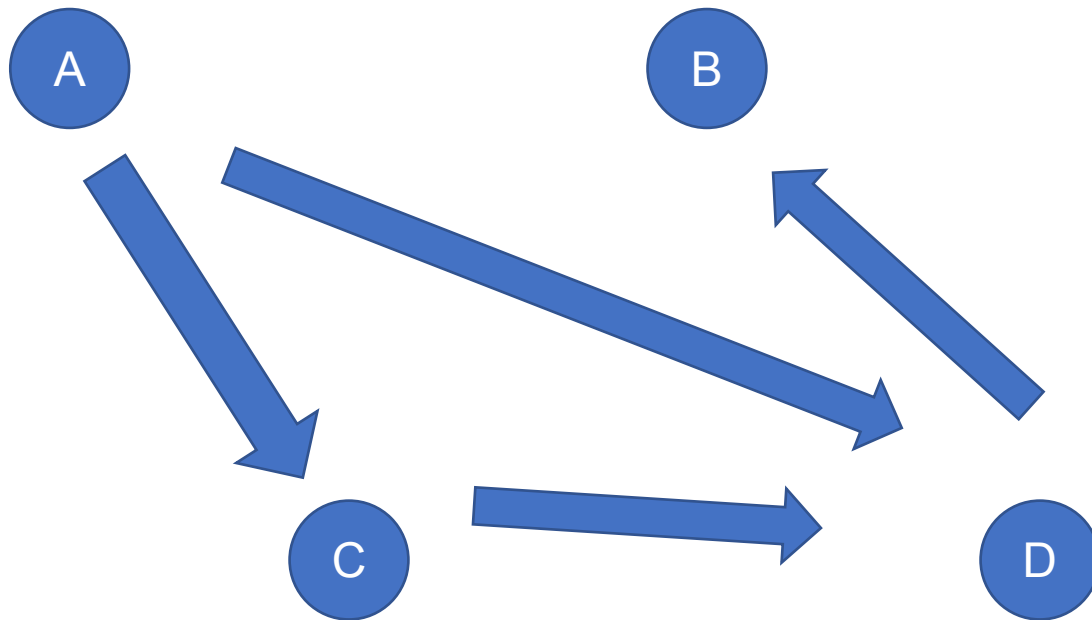
- Join to combine data from different tables





# Recap – Self Joins

- Join to combine data from different tables



# Goals for Today

- We have started to build our SQL toolbox
  - Not just reading and filtering data anymore
  - Starting to answer complex questions
- Today we want to effectively summarize results

# Aggregation functions

- New class of SQL queries

## Aggregates

# Outline

- Aggregation functions
- GROUP BY and HAVING clauses in SQL
- The witnessing problem

# Actionable Results

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**

# Actionable Results

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**
  - “How popular is this tv-show?”

# Actionable Results

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**
  - “How popular is this tv-show?” □ COUNT

# Actionable Results

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**
  - “How popular is this tv-show?” ☐ COUNT
  - “Do I spend too much on coffee?”



# Actionable Results

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**
  - “How popular is this tv-show?” □ COUNT
  - “Do I spend too much on coffee?” □ SUM

# Actionable Results

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**
  - “How popular is this tv-show?” □ COUNT
  - “Do I spend too much on coffee?” □ SUM
  - “Am I being ripped off by this car dealership?”

# Actionable Results

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**
  - “How popular is this tv-show?” □ COUNT
  - “Do I spend too much on coffee?” □ SUM
  - “Am I being ripped off by this car dealership?” □ AVG

# Actionable Results

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**
  - “How popular is this tv-show?” □ COUNT
  - “Do I spend too much on coffee?” □ SUM
  - “Am I being ripped off by this car dealership?” □ AVG
  - “Who got the highest grade in the class?”

# Actionable Results

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**
  - “How popular is this tv-show?” □ COUNT
  - “Do I spend too much on coffee?” □ SUM
  - “Am I being ripped off by this car dealership?” □ AVG
  - “Who got the highest grade in the class?” □ MAX

# Actionable Results

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**
  - “How popular is this tv-show?” □ COUNT
  - “Do I spend too much on coffee?” □ SUM
  - “Am I being ripped off by this car dealership?” □ AVG
  - “Who got the highest grade in the class?” □ MAX
  - “What’s the cheapest food on the Ave?”

# Actionable Results

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**
  - “How popular is this tv-show?” □ COUNT
  - “Do I spend too much on coffee?” □ SUM
  - “Am I being ripped off by this car dealership?” □ AVG
  - “Who got the highest grade in the class?” □ MAX
  - “What’s the cheapest food on the Ave?” □ MIN

# Actionable Results

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**
  - COUNT
  - SUM
  - AVG
  - MAX
  - MIN



# Actionable Results

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**
  - SELECT **COUNT**(\*) FROM StreamingViews ...
  - SELECT **SUM**(cost) FROM CoffeeReceipts ...
  - SELECT **AVG**(price) FROM CarDealers ...
  - SELECT **MAX**(score) FROM StudentGrades ...
  - SELECT **MIN**(price) FROM AveLunchPrices ...

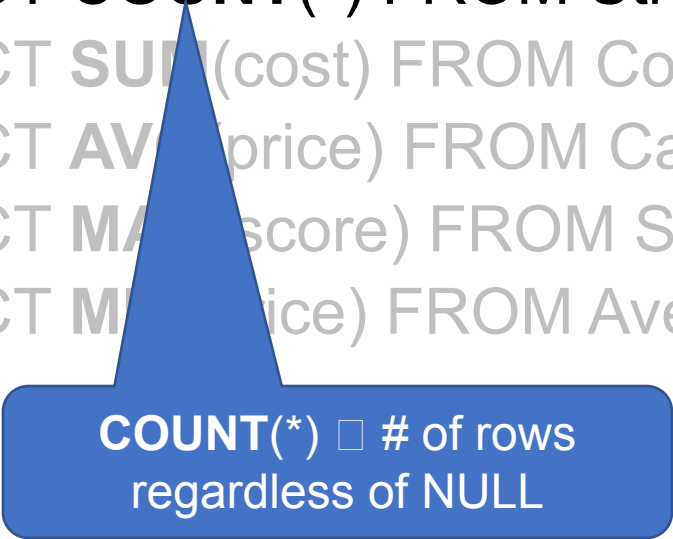


**AGG**(attr) □ computes **AGG** over non-NULL values  
**AGG**(DISTINCT attr) is also possible

# Actionable Results

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**

- **SELECT COUNT(\*)** FROM StreamingViews ...
- SELECT SUM(cost) FROM CoffeeReceipts ...
- SELECT AVG(price) FROM CarDealers ...
- SELECT MAX(score) FROM StudentGrades ...
- SELECT MIN(price) FROM AveLunchPrices ...



**COUNT(\*)** □ # of rows  
regardless of NULL

# Example 1

What is the average salary of people in our dataset?

**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

# Example 1

What is the average salary of people in our dataset?

```
SELECT AVG(P.Salary)
FROM Payroll AS P;
```

**AVG(P.Salary)**

75000

**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

# Example 2

How many professors are in the dataset?

**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

# Example 2

How many professors are in the dataset?

```
SELECT COUNT (*)  
  FROM Payroll AS P  
 WHERE P.Job = 'Prof';
```

COUNT(\*)

2

Aggregate happens after  
WHERE statement

**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

# Example 3

How many TAs are there and what is their average salary?

**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

# Example 3

How many TAs are there and what is their average salary?

```
SELECT COUNT(*), AVG(P.Salary)
FROM Payroll AS P
WHERE P.Job = 'TA';
```

COUNT(*)	AVG(P.Salary)
2	55000

**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



# Example 3

How many TAs are there and what is their average salary?

```
SELECT COUNT(*) AS Num_TAs,  
        AVG(P.Salary) AS Avg_Salary  
FROM Payroll AS P  
WHERE P.Job = 'TA';
```

Num_TAs	Avg_Salary
2	55000

**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

# Aggregation Semantics

What am I aggregating over in a  
SELECT-FROM-WHERE query with joins?

Intuitively: “Everything after the FROM/WHERE”



What does “after FROM/WHERE”  
mean here?

# Aggregation Semantics

What am I aggregating over in a  
SELECT-FROM-WHERE query with joins?

Intuitively: “Everything after the FROM/WHERE”



What does “after FROM/WHERE”  
mean here?

In SQL, the joins and where statements produce one **intermediate relation**, after which the aggregate is calculated.

# Aggregation Semantics

Will this query get me the correct calculation for average salary of all people who own cars?

```
SELECT AVG(P.Salary)
FROM Payroll AS P, Regist AS R
WHERE P.UserID = R.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

# Aggregation Semantics

Will this query get me the correct calculation for average salary of all people who own cars?

```
SELECT AVG(P.Salary)
FROM Payroll AS P, Regist AS R
WHERE P.UserID = R.UserID;
```

Should be 70,000, let's find out if that's what we get

**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

# Aggregation Semantics

```
SELECT AVG(P.Salary)
FROM Payroll AS P, Regist AS R
WHERE P.UserID = R.UserID;
```

$\text{Join}_{P.UserID=R.UserID}$

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

UserID	Car
123	Charger
567	Civic
567	Pinto



# Aggregation Semantics

```
SELECT AVG(P.Salary)
FROM Payroll AS P, Regist AS R
WHERE P.UserID = R.UserID;
```

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

$\text{Join}_{P.UserID=R.UserID}$

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

UserID	Car
123	Charger
567	Civic
567	Pinto



# Aggregation Semantics

```
SELECT AVG(P.Salary)
  FROM Payroll AS P, Regist AS R
 WHERE P.UserID = R.UserID;
```

Aggregate<sub>AVG(P.Salary)</sub>



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

Join<sub>P.UserID=R.UserID</sub>

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

UserID	Car
123	Charger
567	Civic
567	Pinto





# Aggregation Semantics

```
SELECT AVG(P.Salary)
  FROM Payroll AS P, Regist AS R
 WHERE P.UserID = R.UserID;
```

Aggregate<sub>AVG(P.Salary)</sub>




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

Join<sub>P.UserID=R.UserID</sub>

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

UserID	Car
123	Charger
567	Civic
567	Pinto



# Aggregation Semantics

```
SELECT AVG(P.Salary)
  FROM Payroll AS P, Regist AS R
 WHERE P.UserID = R.UserID;
```

**AVG(P.Salary)**

76666

Aggregate<sub>AVG(P.Salary)</sub>



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

Join<sub>P.UserID=R.UserID</sub>

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

UserID	Car
123	Charger
567	Civic
567	Pinto



# Aggregation Semantics

```
SELECT AVG(P.Salary)
  FROM Payroll AS P, Regist AS R
 WHERE P.UserID = R.UserID;
```

AVG(P.Salary)

76666

Not 70,000!

Aggregate<sub>AVG(P.Salary)</sub>

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

Join<sub>P.UserID=R.UserID</sub>

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

UserID	Car
123	Charger
567	Civic
567	Pinto

# Aggregation Semantics

```
SELECT AVG(P.Salary)
  FROM Payroll AS P, Regist AS R
 WHERE P.UserID = R.UserID;
```

**AVG(P.Salary)**

76666

90000 was counted  
twice...

Aggregate  $AVG(P.Salary)$

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

Join  $P.UserID = R.UserID$

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

UserID	Car
123	Charger
567	Civic
567	Pinto

# Aggregation Semantics

Compute the average salary of all people who own cars?

(Did not work, need subqueries for this)

Compute the **minimum** salary of all people who own cars?

(This will work, check for yourself!)

```
SELECT MIN(P.Salary)  
FROM Payroll AS P, Regist AS R  
WHERE P.UserID = R.UserID;
```

**Payroll**

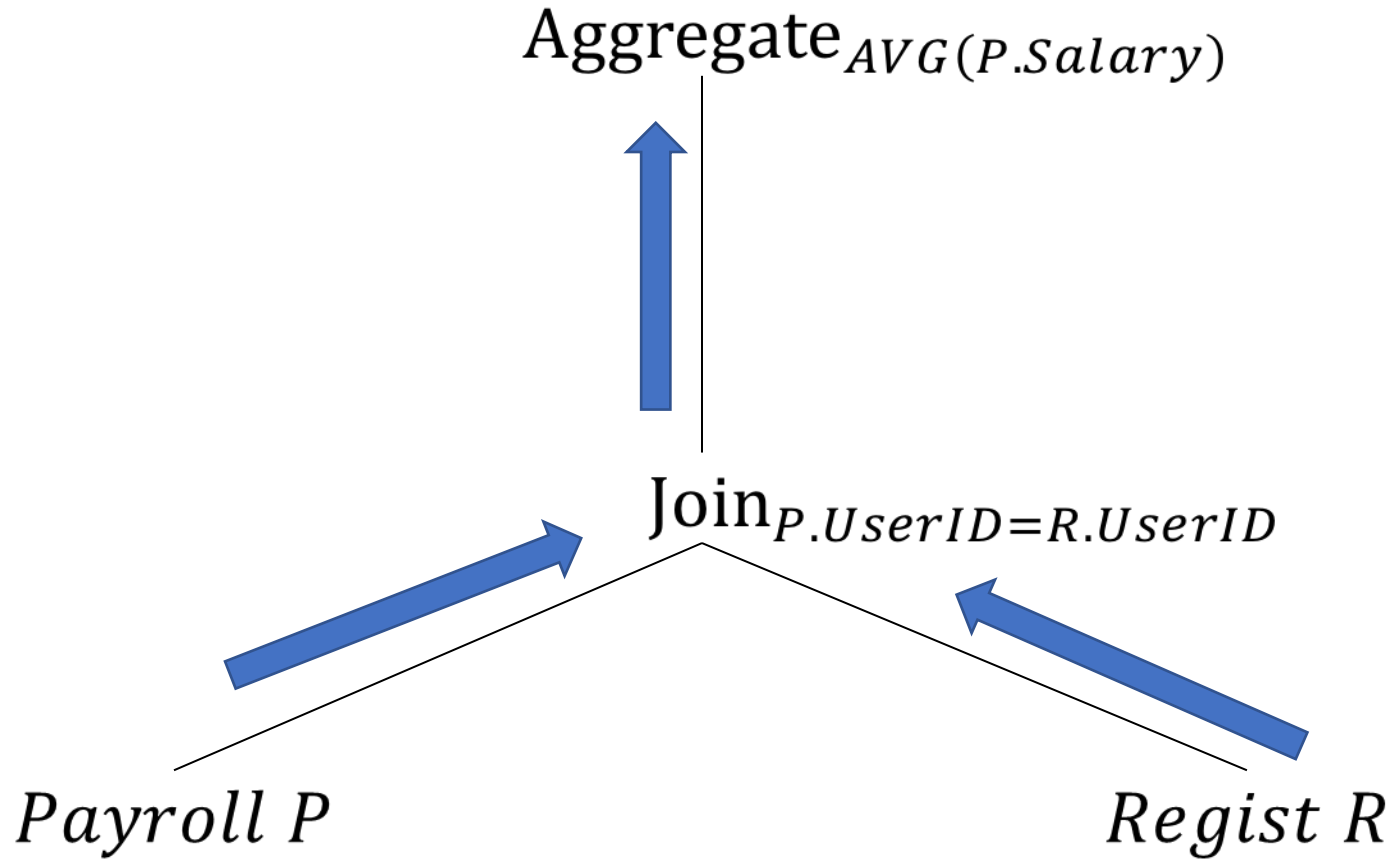
UserID	Name	Job	Salary
123	Jack	TA	50000
345	Allison	TA	60000
567	Magda	Prof	90000
789	Dan	Prof	100000

**Regist**

UserID	Car
123	Charger
567	Civic
567	Pinto

# Aggregation Semantics

```
SELECT AVG(P.Salary)
FROM Payroll AS P, Regist AS R
WHERE P.UserID = R.UserID;
```



# Aggregation Semantics

```
SELECT AVG(P.Salary)
```

```
FROM Payroll AS P, Regist AS R
```

```
WHERE P.UserID = R.UserID;
```

AVG(P.Salary)
76666

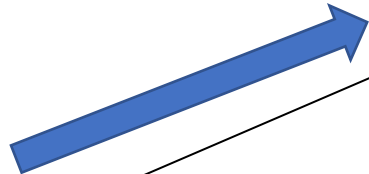
Aggregate<sub>AVG(P.Salary)</sub>

**Aggregation**

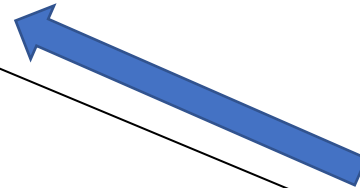
Does not keep other columns than the aggregate fields



Join<sub>P.UserID=R.UserID</sub>



*Payroll P*



*Regist R*

# Grouping

- SQL allows you to specify what groups your query operates over
  - Sometimes a “whole-table” aggregation is too coarse-grained
  - We can partition our data based on **matching attribute values**



# Grouping

- SQL allows you to specify what groups your query operates over
  - Sometimes a “whole-table” aggregation is too coarse-grained
  - We can partition our data based on **matching attribute values**

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

...

**GROUP BY** Job

...

# Grouping

- SQL allows you to specify what groups your query operates over
  - Sometimes a “whole-table” aggregation is too coarse-grained
  - We can partition our data based on **matching attribute values**

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

...

**GROUP BY** Job

...

# Grouping Example

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

# Grouping Example

```
SELECT Job, MAX(Salary)
FROM Payroll
GROUP BY Job
```

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

Job	MAX(Salary)
TA	60000
Prof	100000

# Grouping on Multiple Attributes

```
SELECT Name, MAX(Salary)
FROM Payroll
GROUP BY Job, Name
```

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

Name	Salary
Jack	50000
Allison	60000
Magda	90000
Dan	100000

# Filtering Groups with HAVING

```
SELECT Job, MAX(Salary)
FROM Payroll
GROUP BY Job
HAVING MIN(Salary) > 80000
```

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

# Filtering Groups with HAVING

```
SELECT Job, MAX(Salary)
FROM Payroll
GROUP BY Job
HAVING MIN(Salary) > 80000
```

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

Job	MAX(Salary)
Prof	100000

# Aggregation RA

How is aggregation processed internally?

```
SELECT Job, MAX(Salary)
FROM Payroll
GROUP BY Job
HAVING MIN(Salary) > 80000
```



# Aggregation RA

How is aggregation processed internally?

```
SELECT Job, MAX(Salary)
FROM Payroll
GROUP BY Job
HAVING MIN(Salary) > 80000
```

Our first preview of Relational Algebra:

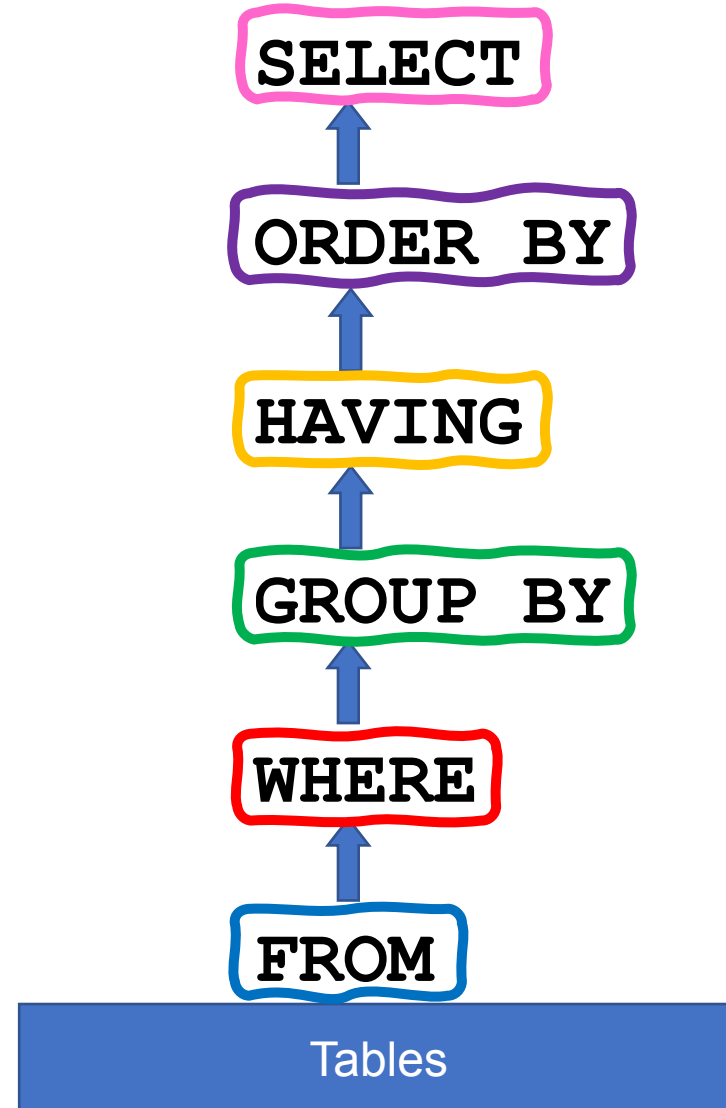
“Having” applies **after** grouping the big intermediate table

# SQL and RA Vocab Summary

## FWGHOS

TM

**SELECT** ...  
**FROM** ...  
**WHERE** ...  
**GROUP BY** ...  
**HAVING** ...  
**ORDER BY** ...



# Aggregation RA

```
SELECT Job, MAX(Salary)
FROM Payroll
GROUP BY Job
HAVING MIN(Salary) > 80000
```

UserID	Name	Job	Salary
...	...	...	...

# Aggregation RA

```
SELECT Job, MAX(Salary)
  FROM Payroll
 GROUP BY Job
HAVING MIN(Salary) > 80000
```

Aggregate  $Job, MAX(P.Salary) \rightarrow maxSal, MIN(P.Salary) \rightarrow minSal$

UserID	Name	Job	Salary
...	...	...	...

# Aggregation RA

```
SELECT Job, MAX(Salary)
  FROM Payroll
 GROUP BY Job
HAVING MIN(Salary) > 80000
```

Job	maxSal	minSal
TA	60000	50000
Prof	100000	90000

Aggregate  $Job, MAX(P.Salary) \rightarrow maxSal, MIN(P.Salary) \rightarrow minSal$

UserID	Name	Job	Salary
...	...	...	...

# Aggregation RA

```
SELECT Job, MAX(Salary)
  FROM Payroll
 GROUP BY Job
HAVING MIN(Salary) > 80000
```

Job	maxSal	minSal
Prof	100000	90000

Having  $minSal > 80000$

Job	maxSal	minSal
TA	60000	50000
Prof	100000	90000

Aggregate  $Job, MAX(P.Salary) \rightarrow maxSal, MIN(P.Salary) \rightarrow minSal$

UserID	Name	Job	Salary
...	...	...	...

# Aggregation RA

```
SELECT Job, MAX(Salary)
  FROM Payroll
 GROUP BY Job
HAVING MIN(Salary) > 80000
```

Select<sub>*Job, maxSal*</sub>

Job	maxSal	minSal
Prof	100000	90000

Having<sub>*minSal > 80000*</sub>

Job	maxSal	minSal
TA	60000	50000
Prof	100000	90000

Aggregate<sub>*Job, MAX(P.Salary) → maxSal, MIN(P.Salary) → minSal*</sub>

UserID	Name	Job	Salary
...	...	...	...

# Aggregation RA

```
SELECT Job, MAX(Salary)
  FROM Payroll
 GROUP BY Job
HAVING MIN(Salary) > 80000
```

Job	maxSal
Prof	100000

Select<sub>*Job, maxSal*</sub>

Job	maxSal	minSal
Prof	100000	90000

Having<sub>*minSal > 80000*</sub>

Job	maxSal	minSal
TA	60000	50000
Prof	100000	90000

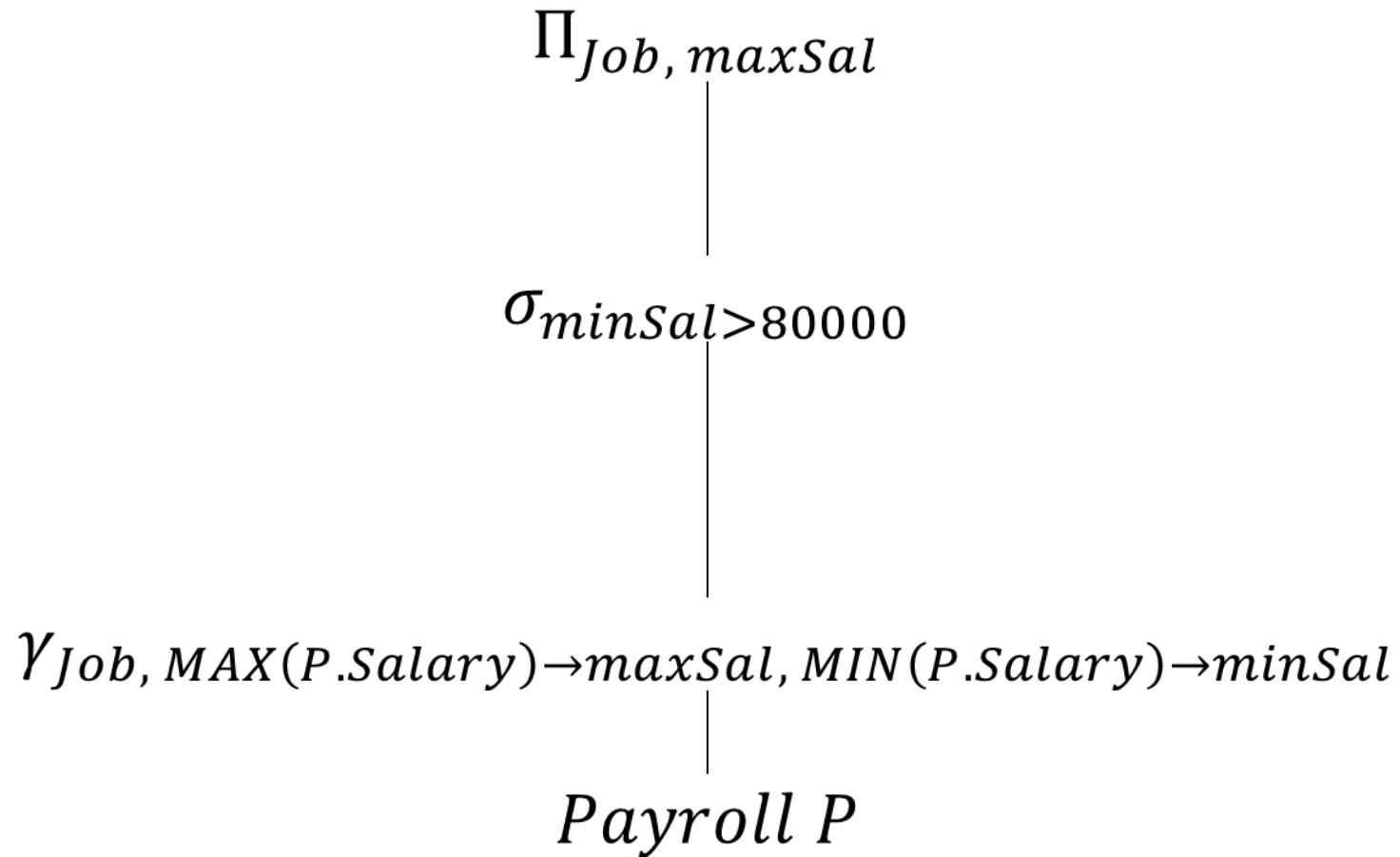
Aggregate<sub>*Job, MAX(P.Salary) → maxSal, MIN(P.Salary) → minSal*</sub>

UserID	Name	Job	Salary
...	...	...	...



# Aggregation RA

```
SELECT Job, MAX(Salary)
FROM Payroll
GROUP BY Job
HAVING MIN(Salary) > 80000
```



# Aggregation RA

```
SELECT Job, MAX(Salary)
FROM Payroll
GROUP BY Job
HAVING MIN(Salary) > 80000
```

## Selection

HAVING uses the same symbol and operation used by WHERE clause

$\Pi_{Job, maxSal}$

$\sigma_{minSal > 80000}$

$\gamma_{Job, MAX(P.Salary) \rightarrow maxSal, MIN(P.Salary) \rightarrow minSal}$

*Payroll P*

# Preview: Relational Algebra

**SELECT** ...  
**FROM** ...  
**WHERE** ...  
**GROUP BY** ...  
**HAVING** ...  
**ORDER BY** ...

$\delta$

$\Pi$

$\tau$

$\sigma$

$\gamma$

$\sigma \bowtie \times \dots$

Tables

# SQL and RA Vocab Summary

**SELECT** ...  
**FROM** ...  
**WHERE** ...  
**GROUP BY** ...  
**HAVING** ...  
**ORDER BY** ...

$\delta$

$\Pi$

$\tau$

$\sigma$

$\gamma$

Selection  
Join  
Cartesian Product

$\sigma \bowtie \times \dots$

Tables

# SQL and RA Vocab Summary

**SELECT** ...  
**FROM** ...  
**WHERE** ...  
**GROUP BY** ...  
**HAVING** ...  
**ORDER BY** ...

$\delta$

$\Pi$

$\tau$

$\sigma$

$\gamma$

Aggregation

$\sigma \bowtie \times \dots$

Tables

# SQL and RA Vocab Summary

**SELECT** ...  
**FROM** ...  
**WHERE** ...  
**GROUP BY** ...  
**HAVING** ...  
**ORDER BY** ...

$\delta$

$\Pi$

$\tau$

$\sigma$

Selection

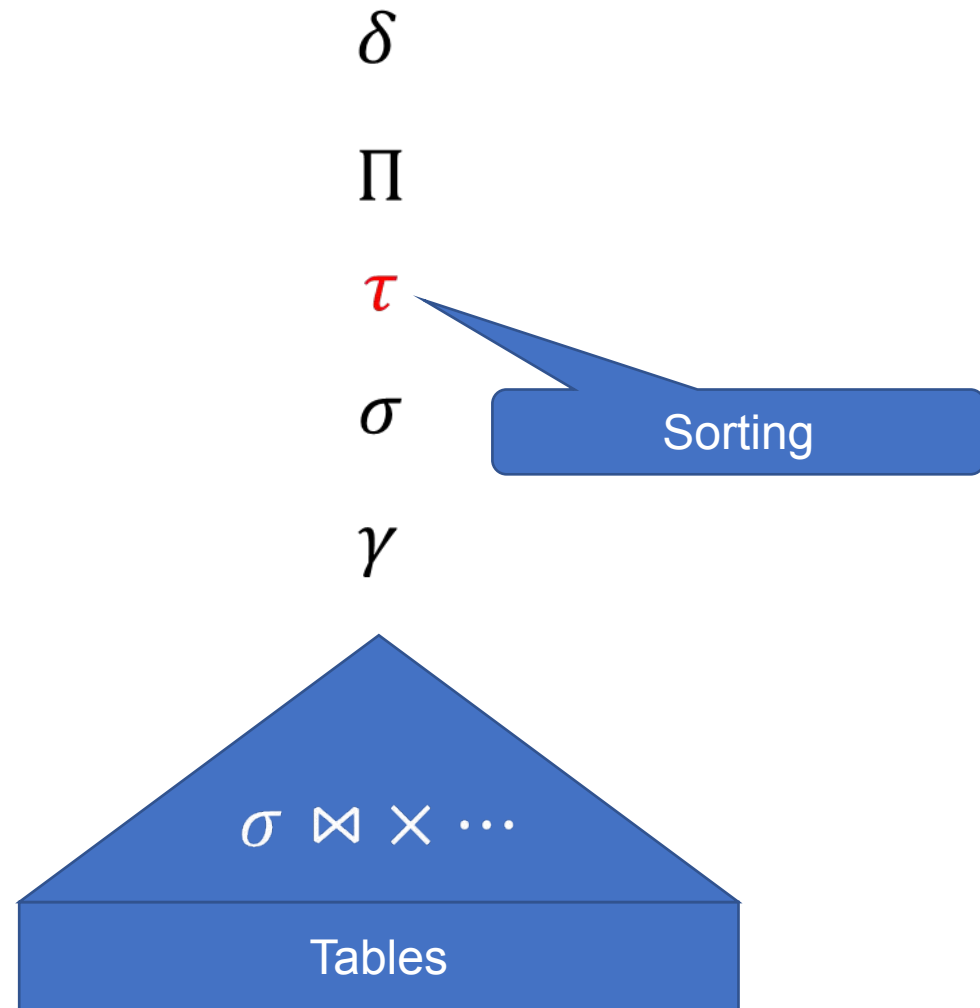
$\gamma$

$\sigma \bowtie \times \dots$

Tables

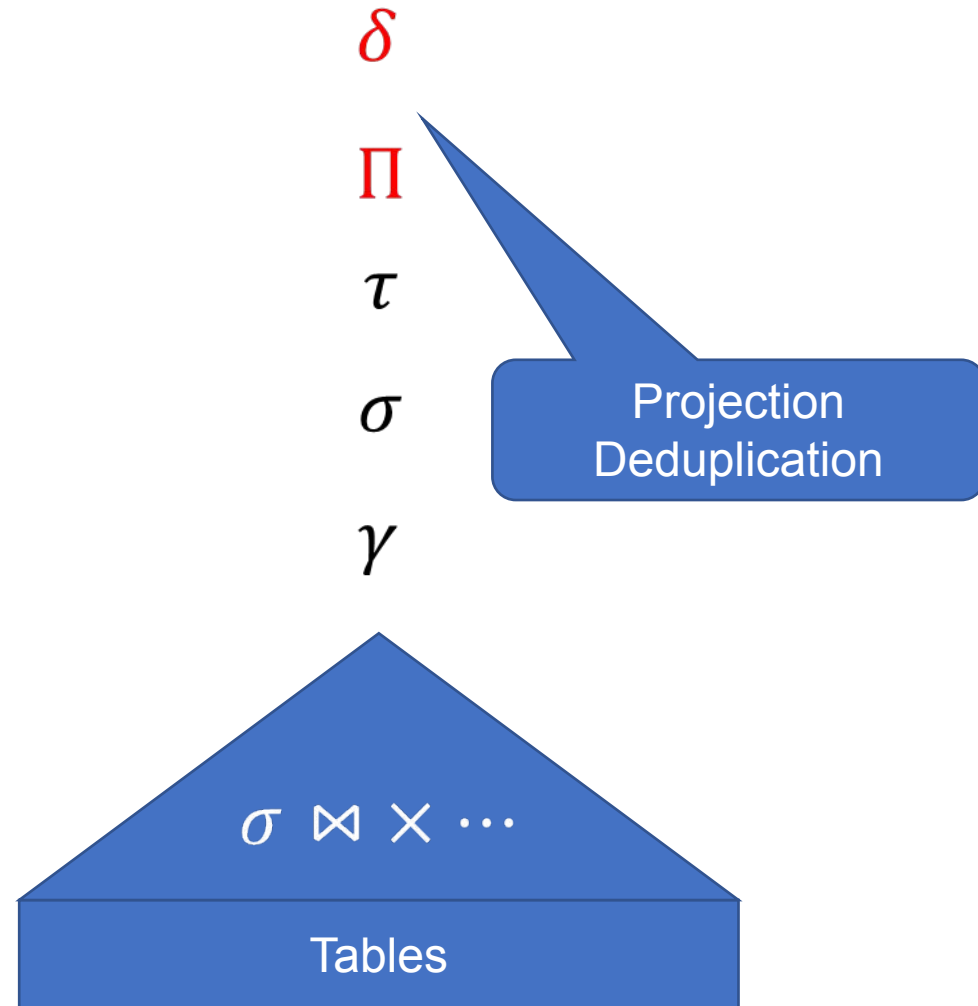
# SQL and RA Vocab Summary

**SELECT** ...  
**FROM** ...  
**WHERE** ...  
**GROUP BY** ...  
**HAVING** ...  
**ORDER BY** ...



# SQL and RA Vocab Summary

**SELECT** ...  
**FROM** ...  
**WHERE** ...  
**GROUP BY** ...  
**HAVING** ...  
**ORDER BY** ...





# SQL and RA Vocab Summary

## FWGHOS

**SELECT** ...  
**FROM** ...  
**WHERE** ...  
**GROUP BY** ...  
**HAVING** ...  
**ORDER BY** ...

$\delta$

$\Pi$

$\tau$

$\sigma$

$\gamma$

$\sigma \bowtie \times \dots$

Tables