

SQL Aggregate Queries

CS430/630
Lecture 8

Aggregate Operators

Significant extension of relational algebra

```
COUNT (*)  
COUNT ( [DISTINCT] A)  
SUM ( [DISTINCT] A)  
AVG ( [DISTINCT] A)  
MAX (A)  
MIN (A)
```

A is a single column

Result is **single** value obtained by applying aggregate over all qualifying tuples

```
SELECT COUNT (*)  
FROM Sailors S
```



Aggregate Queries Examples

```
SELECT AVG (S.age)
FROM Sailors S
WHERE S.rating=10
```

```
SELECT COUNT (DISTINCT S.rating)
FROM Sailors S
WHERE S.sname='Bob'
```

```
SELECT S.sname
FROM Sailors S
WHERE S.rating= (SELECT MAX(S2.rating)
                 FROM Sailors S2)
```

Aggregate + nested!



Common Mistake with Aggregates

```
SELECT S.sname, MAX (S.age)
FROM Sailors S
```

Illegal Query!

- ▶ Can't have both aggregates and non-aggregates in SELECT
 - ▶ Exception: GROUP BY (later in this class)
- ▶ Reason: it is not guaranteed that there is only one tuple with the MAX value



Grouping Results

- ▶ So far, aggregates applied to all (qualifying) tuples
 - ▶ We may want to apply them to each of several groups
- ▶ *“Find the age of the youngest sailor **for each** rating level”*
 - ▶ In general, we don’t know how many rating levels exist, and what the rating values for these levels are!
 - ▶ Suppose we know that rating values go from 1 to 10

```
SELECT MIN (S.age)
FROM Sailors S
WHERE S.rating = 1
```

...

```
SELECT MIN (S.age)
FROM Sailors S
WHERE S.rating = 10
```

```
SELECT MIN (S.age)
FROM Sailors S
WHERE S.rating = 2
```

How to achieve this?



Queries With GROUP BY and HAVING

SELECT	[DISTINCT] <i>target-list</i>
FROM	<i>relation-list</i>
WHERE	<i>qualification</i>
GROUP BY	<i>grouping-list</i>
HAVING	<i>group-qualification</i>

- ▶ The *target-list* contains:
 - (i) attribute names list
 - (ii) terms with aggregate operations (e.g., **MIN** (*S.age*))
- ▶ The attribute list (i) must be a subset of *grouping-list*
 - ▶ A *group* is a set of tuples that have the same value for all attributes in *grouping-list*
 - ▶ Each answer tuple corresponds to a *group*, so these attributes must have a single value per group.



Conceptual Evaluation

1. Compute cross-product of *relation-list*
2. Discard tuples that fail *qualification*, 'unnecessary' fields are deleted
3. Remaining tuples are partitioned into groups by the value of attributes in *grouping-list*
4. Discard groups that fail *group-qualification*
 - ▶ Expressions in *group-qualification* must have a single value per group!
 - ▶ An attribute in *group-qualification* that is not an argument of an aggregate operation must appear in *grouping-list* (unless **EVERY** or **ANY** used)
5. Generate single answer tuple per qualifying group



GROUPBY Query Example

“Find age of the youngest sailor with age at least 18, for each rating with at least 2 such sailors”

Sailors

<u>sid</u>	sname	rating	age
22	dustin	7	45.0
29	brutus	1	33.0
31	lubber	8	55.5
32	andy	8	25.5
58	rusty	10	35.0
64	horatio	7	35.0
71	zorba	10	16.0
74	horatio	9	35.0
85	art	3	25.5
95	bob	3	63.5
96	frodo	3	25.5

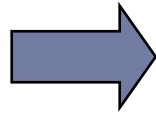
```
SELECT S.rating, MIN (S.age)
           AS minage
FROM Sailors S
WHERE S.age >= 18
GROUP BY S.rating
HAVING COUNT (*) > 1
```



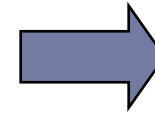
GROUPBY Conceptual Evaluation Example

“Find age of the youngest sailor with age at least 18, for each rating with at least 2 such sailors”

rating	age
7	45.0
1	33.0
8	55.5
8	25.5
10	35.0
7	35.0
10	16.0
9	35.0
3	25.5
3	63.5
3	25.5



rating	age
1	33.0
3	25.5
3	63.5
3	25.5
7	45.0
7	35.0
8	55.5
8	25.5
9	35.0
10	35.0



rating	minage
3	25.5
7	35.0
8	25.5



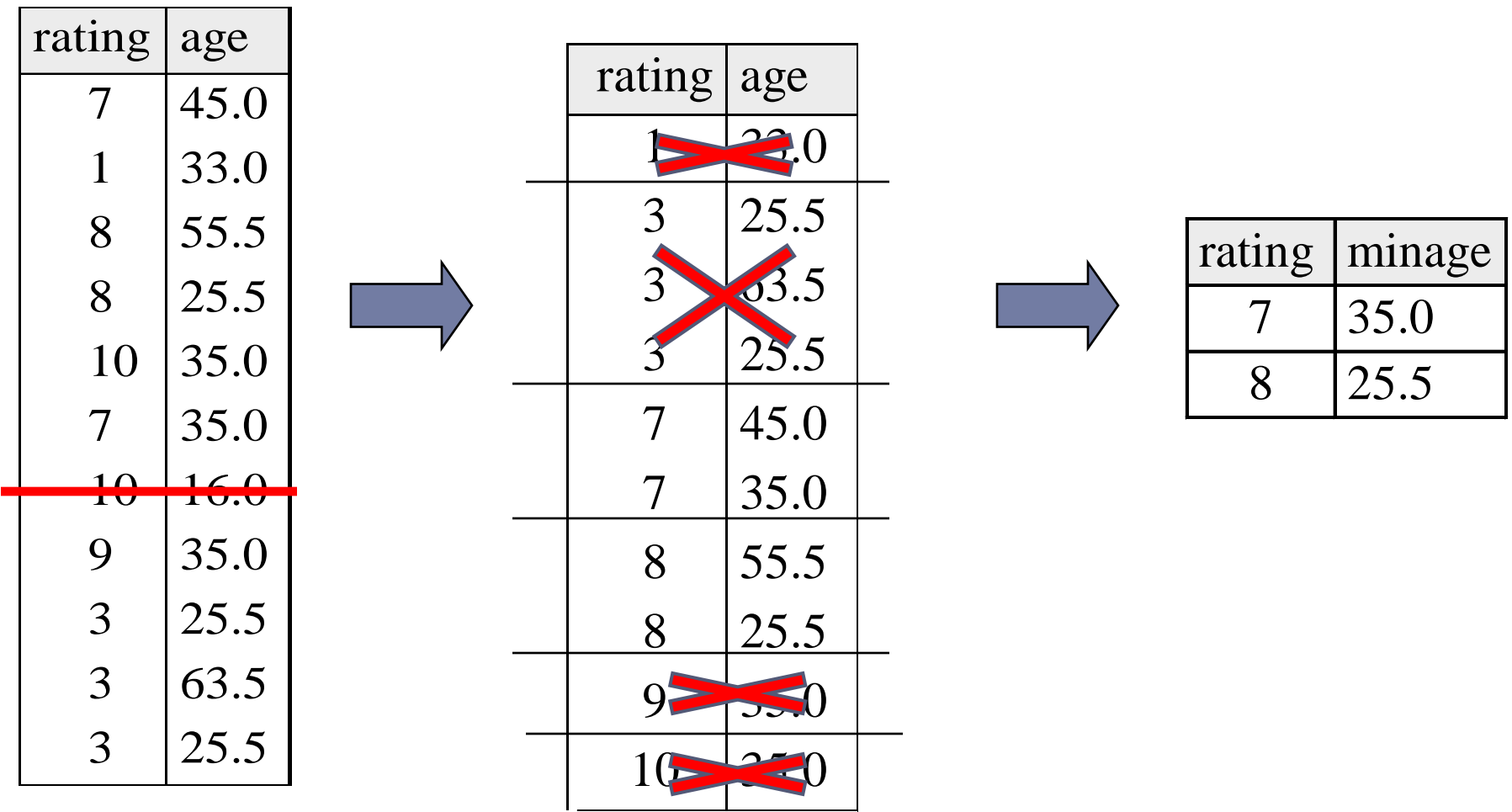
More Group Qualification Functions

- ▶ So far, we have seen group qualification based on a **property of the group**
 - ▶ E.g., aggregate function computed for entire group
- ▶ But recent SQL standard version allow group qualification based on a **property of individual records**
 - ▶ **EVERY(condition)**:TRUE if condition holds for every group tuple
 - ▶ **ANY(condition)**:TRUE if condition holds for some group tuple



Find age of the youngest sailor with age ≥ 18 , for each rating with at least 2 such sailors and with every sailor under 60.

HAVING COUNT (*) > 1 AND **EVERY** (S.age <=60)



Pay attention to order of steps!

▶ **HAVING** executes **AFTER** **WHERE**

*“Find age of the youngest sailor with age ≥ 18 ,
for each rating with at least 2 sailors (of any age)”*

```
SELECT S.rating, MIN (S.age)
FROM Sailors S
WHERE S.age  $\geq$  18
GROUP BY S.rating
HAVING COUNT (*) > 1
```

WRONG!!!



Find age of the youngest sailor with age ≥ 18 ,
for each rating with at least 2 sailors (of any age)

rating	age
7	45.0
1	33.0
8	55.5
8	25.5
10	35.0
7	35.0
10	16.0
9	35.0
3	25.5
3	63.5
3	25.5

rating	age
7	45.0
1	33.0
8	55.5
8	25.5
10	35.0
7	35.0
10	16.0
9	35.0
3	25.5
3	63.5
3	25.5

rating	age
1	33.0
3	25.5
3	63.5
3	25.5
7	45.0
7	35.0
8	55.5
8	25.5
9	35.0
10	35.0

rating	minage
3	25.5
7	35.0
8	25.5
10	35.0



Pay attention to order of steps!

*“Find age of the youngest sailor with age ≥ 18 ,
for each rating with at least 2 sailors (of any age)”*

```
SELECT S.rating, MIN (S.age)
FROM Sailors S
WHERE S.age  $\geq$  18
GROUP BY S.rating
HAVING 1 < (SELECT COUNT (*)
            FROM Sailors S2
            WHERE S.rating=S2.rating)
```

- ▶ **HAVING** executes **AFTER** **WHERE**
- ▶ **HAVING** clause can also contain a subquery!



“Summary of cases” – INFORMAL!

- ▶ Can group validation condition be evaluated on “intermediate” relation alone?
 - ▶ If **NO**, then we need subquery in HAVING
 - ▶ If **YES**, then we do not need subquery, and we have two further cases:
 - ▶ Group validation condition **DOES NOT** depend on individual tuples in group, only aggregates and group-by attributes appear in the HAVING clause
 - ▶ Group validation **DOES** depend on individual tuples in group, in which case non-group-by attributes may appear with ANY or EVERY operator
- ▶ **Note: this is just a guideline for most cases, it is actually possible to have a mix of the above!!!**



Aggregates and FROM Subqueries

- ▶ Aggregate operations cannot be nested!

“Find rating that has lowest average sailor age”

```
SELECT S.rating  
FROM Sailors S  
WHERE S.age = (SELECT MIN (AVG (S2.age)) FROM Sailors S2)
```

WRONG

Correct solution:

```
SELECT Temp.rating, Temp.avgage  
FROM (SELECT S.rating, AVG (S.age) AS avgage  
      FROM Sailors S  
      GROUP BY S.rating) Temp  
WHERE Temp.avgage = (SELECT MIN (Temp.avgage)  
                    FROM Temp)
```

