



# CS 564: Database Management Systems

## Lecture 4: Advanced SQL I

Xiangyao Yu

1/31/2024

# Module A1: SQL

---

SQL: Basics I

SQL: Basics II

**Advanced SQL I**

- Aggregation, nulls, and outer joins

Advanced SQL II

# Outline of this Lecture

---

## Aggregation and Group By

- Aggregate
- GROUP BY
- HAVING

## Outer Joins

## Null Values

# Example Database

**Sailors** (sid: integer, **sname**: string, **rating**: integer, **age**: real)

**Boats** (bid: integer, **bname**: string, **color**: string)

**Reserves** (sid: integer, bid: integer, **day**: date)

**Sailors**

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35
85	Art	3	25.5
95	Bob	3	63.5

**Boats**

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

**Reserves**

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

# Aggregation (Recap)

**SUM, AVG, COUNT, MIN, MAX** can be applied to a column in a **SELECT** clause to produce that aggregation on the column

**COUNT(\*)** simply counts the number of tuples

```
SELECT AVG(age)
FROM Sailors
WHERE rating > 4;
```

**Sailors**

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35
85	Art	3	25.5
95	Bob	3	63.5

**Reserves**

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

**Boats**

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

# GROUP BY

---

We may follow a **SELECT-FROM-WHERE** expression by **GROUP BY** and a list of attributes

The relation is then grouped according to the values of those attributes, and any aggregation is applied only **within each group**

```
SELECT COUNT(*), S.rating  
FROM Sailors S  
GROUP BY S.rating;
```

# GROUP BY: Example

```
SELECT A, SUM(B * C)
FROM R
GROUP BY A;
```

**R**

A	B	C
a	2	0
a	5	1
b	7	1
b	6	0
c	4	1

grouping

A	B	C
a	2	0
	5	1
b	7	1
	6	0
c	4	1

SELECT  
clause

$$5 = 2*0 + 5*1$$

A	SUM(B*C)
a	5
b	7
c	4

# Restrictions

---

If any aggregation is used, then each element of the **SELECT** list must be either:

- aggregated, or
- an attribute on the **GROUP BY** list



# Restrictions

---

If any aggregation is used, then each element of the **SELECT** list must be either:

- aggregated, or
- an attribute on the **GROUP BY** list

This query is **wrong!!**

```
SELECT S.sname, AVG(S.age)
FROM Sailors S
GROUP BY rating;
```

# GROUP BY – Example

For each red boat, find the number of reservations for this boat

**Sailors**

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35
85	Art	3	25.5
95	Bob	3	63.5

**Reserves**

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

**Boats**

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

# GROUP BY – Example

For each red boat, find the number of reservations for this boat

```
SELECT  B.bid, count(*) AS reservationcount
FROM    Reserves R, Boats B
WHERE   B.bid = R.bid
        AND   B.color = 'red'
GROUP BY B.bid;
```

**Sailors**

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35
85	Art	3	25.5
95	Bob	3	63.5

**Reserves**

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

**Boats**

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

# GROUP BY + HAVING

---

The **HAVING** clause **always** follows a **GROUP BY** clause in a SQL query

- It applies to each group, and groups not satisfying the condition are removed
- It can refer only to attributes of relations in the **FROM** clause, as long as the attribute makes sense within a group

The HAVING clause applies **only** on aggregates!

# HAVING – Example

Find the average age of sailors for each rating level that has at least two sailors

```
SELECT  S.rating, AVG(S.age) AS avgage
FROM    Sailors S
GROUP BY S.rating
HAVING  count(*) > 1;
```

**Sailors**

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35
85	Art	3	25.5
95	Bob	3	63.5

**Reserves**

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

**Boats**

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

# Incorrect Example

For each red boat, find the number of reservations for this boat

```
SELECT  B.bid, count(*) AS reservationcount
FROM    Reserves R, Boats B
WHERE   B.bid = R.bid
      AND B.color = 'red'
GROUP BY B.bid;
```

**Sailors**

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35
85	Art	3	25.5
95	Bob	3	63.5

**Reserves**

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

**Boats**

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Correct

# Incorrect Example

For each red boat, find the number of reservations for this boat

```
SELECT  B.bid, count(*) AS reservationcount
FROM    Reserves R, Boats B
WHERE   B.bid = R.bid
      AND B.color = 'red'
GROUP BY B.bid;
```

```
SELECT  B.bid, count(*) AS reservationcount
FROM    Reserves R, Boats B
WHERE   B.bid = R.bid
GROUP BY B.bid
HAVING  B.color = 'red';
```

Sailors

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35
85	Art	3	25.5
95	Bob	3	63.5

Reserves

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Boats

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Correct

Incorrect

Columns in the HAVING clause must either (1) appear in the GROUP BY clause, or (2) be an aggregate on a group

# HAVING – Example

Find the average age of sailors for each rating level that has at least two sailors that are at least 18 years old

```
SELECT  S.rating, AVG(S.age) AS avgage
FROM    Sailors S
GROUP BY S.rating
HAVING  1 < (SELECT count(*)
              FROM    Sailors S2
              WHERE   S2.age >= 18
              AND     S2.rating = S.rating);
```

**Sailors**

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35
85	Art	3	25.5
95	Bob	3	63.5

**Reserves**

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

**Boats**

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red



# Putting It All Together

---

```
SELECT    [DISTINCT] exp1, exp2, ...  
FROM      R, S, T ,...  
WHERE     C1  
GROUP BY attributes  
HAVING    C2  
ORDER BY attribute ASC/DESC  
LIMIT N;
```

# Conceptual Evaluation

---

1. Compute the **FROM-WHERE** part, obtain a table with all attributes in R,S,T,... (i.e., a cross product)
2. Group the attributes in the **GROUP BY**
3. Compute the aggregates and keep only groups satisfying condition **C2** in the **HAVING** clause
4. Order by the attributes specified in **ORDER BY**
5. Limit the output if necessary

```
SELECT    [DISTINCT] S
FROM      R, S, T ,...
WHERE     C1
GROUP BY  attributes
HAVING    C2
ORDER BY  attribute ASC/DESC
LIMIT N;
```

# Example

Find the top 2 boat colors that are reserved the most number of times

**Sailors**

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35
85	Art	3	25.5
95	Bob	3	63.5

**Reserves**

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

**Boats**

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

# Example

Find the top 2 boat colors that are reserved the most number of times

```
SELECT  B.color, COUNT(*) AS reservationcount
FROM    Reserves R, Boats B
WHERE   R.bid = B.bid
GROUP BY B.color
ORDER BY reservationcount DESC
LIMIT 2;
```

**Sailors**

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35
85	Art	3	25.5
95	Bob	3	63.5

**Reserves**

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

**Boats**

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

# Outline of this Lecture

---

## Aggregation and Group By

- Aggregate
- GROUP BY
- HAVING

## Outer Joins

## Null Values

# Inner Joins

The joins we have seen so far are **inner joins**

```
SELECT  S.sid, count(*) AS reservationcount
FROM    Sailors S, Reserves R
WHERE   S.sid = R.sid
GROUP BY S.sid;
```

Alternative syntax:

```
SELECT  S.sid, count(*) AS reservationcount
FROM    Sailors S
        INNER JOIN Reserves R ON S.sid = R.sid
GROUP BY S.sid;
```

We can simply also write **JOIN**

Sailors

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55

Reserves

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98

Output

sid	reservationcount
22	4
31	3

# Left Outer Join – Example

**R**

A	B
a	2
a	5
b	5
c	6

**S**

B	C
2	100
3	200
5	300
7	400



```
SELECT A, C
FROM R LEFT OUTER JOIN S
ON R.B = S.B
```

A	C
a	100
a	300
b	300
c	NULL

# Left Outer Join

A **left outer join** includes tuples from the left relation even if there's no match on the right! It fills the remaining attributes with NULL

```
SELECT  S.sid, count(R.sid) AS reservationcount
FROM    Sailors S
LEFT OUTER JOIN Reserves R ON S.sid = R.sid
GROUP BY S.sid;
```

Sailors

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55

Reserves

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98

Output

sid	reservationcount
22	4
29	0
31	3



# Other Outer Joins

## Left outer join:

- include the left tuple even if there is no match
- `R LEFT OUTER JOIN S ON R.B=S.B`

## Right outer join:

- include the right tuple even if there is no match
- `R RIGHT OUTER JOIN S ON R.B=S.B`

## Full outer join:

- include both left and right tuples even if there is no match
- `R FULL OUTER JOIN S ON R.B=S.B`

R	A	B	S	B	C		A	C
	a	2		2	100		a	100
	a	5		3	200	→	a	300
	b	5		5	300		b	300
	c	6		7	400		c	NULL

R	A	B	S	B	C		A	C
	a	2		2	100		a	100
	a	5		3	200	→	a	300
	b	5		5	300		b	300
	c	6		7	400		NULL	400
							NULL	200

R	A	B	S	B	C		A	C
	a	2		2	100		a	100
	a	5		3	200	→	a	300
	b	5		5	300		b	300
	c	6		7	400		c	NULL
							NULL	400
							NULL	200

# Outline of this Lecture

---

## Aggregation and Group By

- Aggregate
- GROUP BY
- HAVING

## Outer Joins

## Null Values

# NULL Values

---

Tuples in SQL relations can have **NULL** as a value for one or more attributes

The meaning depends on context:

- **Missing value**: e.g. we know that Greece has some population, but we don't know what it is
- **Inapplicable**: e.g. the value of attribute *spouse* for an unmarried person

# NULL Propagation

---

When we do arithmetic operations using **NULL**, the result is again a **NULL**

- $(10 * x) + 5$  returns **NULL** if  $x = \text{NULL}$
- $\text{NULL}/0$  also returns **NULL**!

String concatenation also results in **NULL** when one of the operands is **NULL**

- `'Wisconsin' || NULL || '-Madison'` returns **NULL**

# Comparisons With NULL

---

The logic of conditions in SQL is **3-valued logic**:

- **TRUE** = 1
- **FALSE** = 0
- **UNKNOWN** = 0.5

When any value is compared with a **NULL**, the result is **UNKNOWN**

- *e.g.*  $x > 5$  is **UNKNOWN** if  $x = \text{NULL}$

A query produces a tuple in the answer **only if** its truth value in the **WHERE** clause is **TRUE** (1)

# 3-Valued Logic

---

The truth value of a **WHERE** clause is computed using the following rules:

- **C1 AND C2** ---->  $\min\{\text{value}(C1), \text{value}(C2)\}$
- **C1 OR C2** ---->  $\max\{\text{value}(C1), \text{value}(C2)\}$
- **NOT C** ---->  $1 - \text{value}(C)$

# 3-Valued Logic – Example

**SELECT \***

**FROM R**

**WHERE (R.A > 0) AND ((R.B < 5) OR (NOT R.C = 3));**

**tuple (1, NULL, NULL)**

1

0.5

0.5

0.5 (1-0.5)

0.5 (max{0.5, 0.5})

0.5 (min{0.5, 1})

**the expression is UNKNOWN!**

# Complication

---

What will happen in the following query?

```
SELECT COUNT(*)  
FROM Country  
WHERE IndepYear > 1990 OR IndepYear <= 1990 ;
```

It will not count the rows with NULL!



# Testing for NULL

---

We can test for **NULL** explicitly:

- x **IS** NULL
- x **IS** NOT NULL

```
SELECT COUNT(*)  
FROM Country  
WHERE IndepYear > 1990 OR IndepYear <= 1990  
OR IndepYear IS NULL;
```

# View (WITH and CREATE VIEW)

---

```
WITH viewname AS (  
    SELECT ...  
    FROM ...  
    WHERE ...  
)  
SELECT ...
```

```
CREATE VIEW viewname AS (  
    SELECT ...  
    FROM ...  
    WHERE ...  
)  
SELECT ...
```

View: a virtual table based on the output set of a SQL query.

A view can be used just like a normal table

# Jupyter Notebook

---

# Summary

---

## SQL: Aggregation

- Aggregate operators
- GROUP BY
- HAVING

## SQL: Nulls

## SQL: Outer Joins