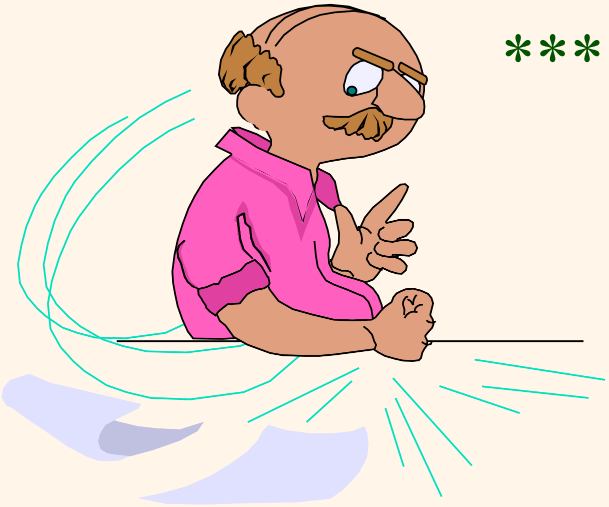
Introduction to Data Management

**** The "Flipped" Edition ****

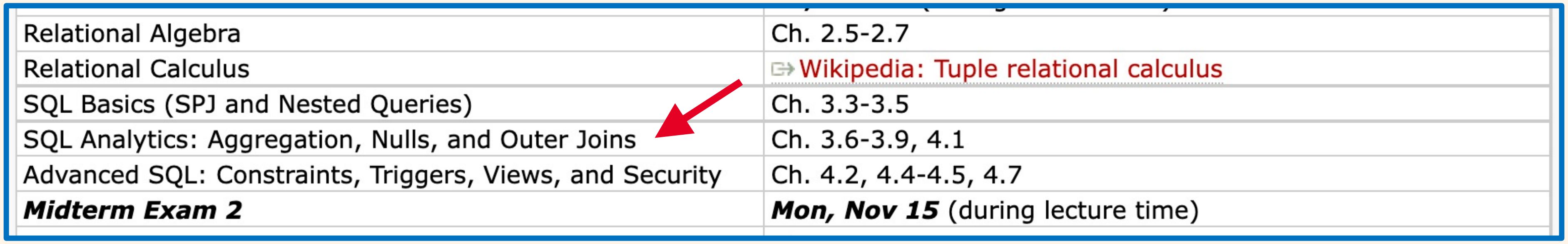


Lecture #14
(SQL III)

Instructor: Mike Carey
mjcarey@ics.uci.edu

Announcements

- You are here:



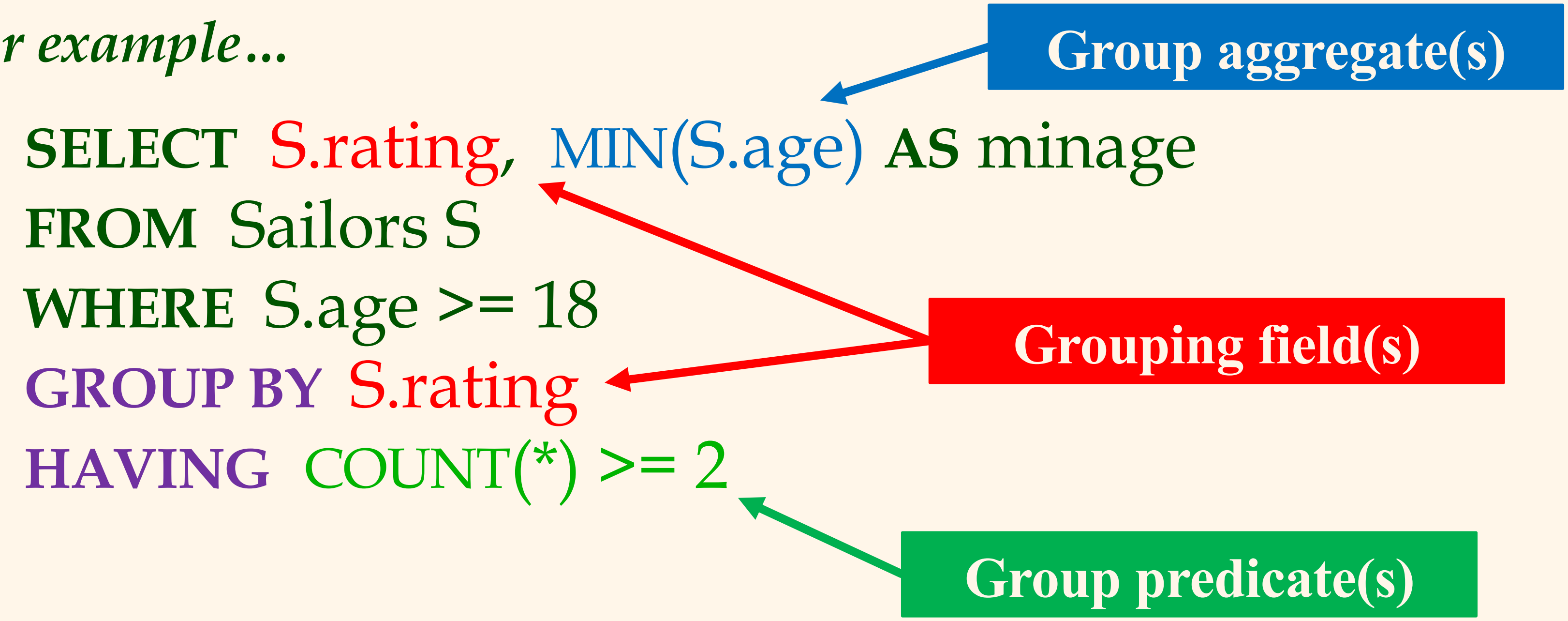
Relational Algebra	Ch. 2.5-2.7
Relational Calculus	⇨Wikipedia: Tuple relational calculus
SQL Basics (SPJ and Nested Queries)	Ch. 3.3-3.5
SQL Analytics: Aggregation, Nulls, and Outer Joins	Ch. 3.6-3.9, 4.1
Advanced SQL: Constraints, Triggers, Views, and Security	Ch. 4.2, 4.4-4.5, 4.7
Midterm Exam 2	***Mon, Nov 15*** (during lecture time)

- HW #4 is in flight…!
  - Due Friday (so you can RelaX a bunch this week… ☺)
  - Be sure to run and check all your queries!
- HW #5 due out Friday (we're in "*Friday mode*" now)
  - First of a series of SQL-based HW assignments
  - It's ***critical*** that you resolve any lingering PostgreSQL issues! (Ask in discussion, post *Q*'s on Piazza, do whatever it takes – otherwise you won't survive…!)

Grouped Aggregation (Review)

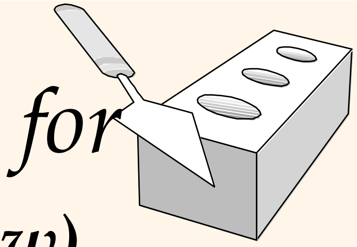
```
SELECT      [DISTINCT]  target-list
FROM        relation-list
WHERE       qualification
GROUP BY    grouping-list
HAVING      group-qualification
```

Our example…



```
SELECT  S.rating,  MIN(S.age) AS minage
FROM  Sailors S
WHERE  S.age >= 18
GROUP BY  S.rating
HAVING  COUNT(*) >= 2
```

Find age of the youngest sailor with age ≥18 for each rating with at least 2 such sailors. **(Review)**



Query:

```
SELECT  S.rating, MIN(S.age) AS minage
FROM  Sailors S
WHERE  S.age >= 18
GROUP BY  S.rating
HAVING  COUNT(*) >= 2
```

Sailors instance:

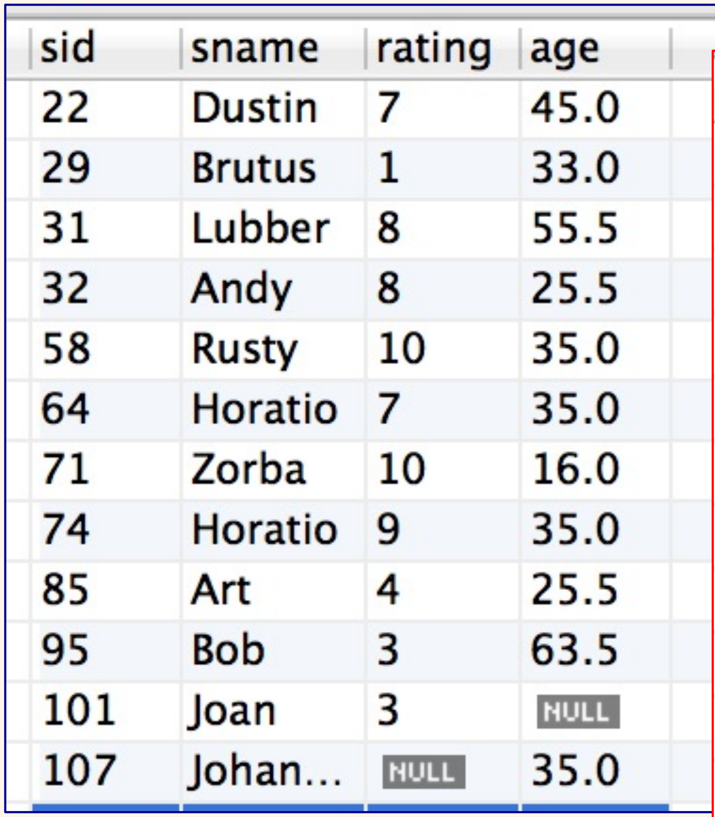
sid	sname	rating	age
22	dustin	7	45.0
29	brutus	1	33.0
31	lubber	8	55.5
32	andy	8	25.5
58	rusty	10	35.0
64	horatio	7	35.0
71	zorba	10	16.0
74	horatio	9	35.0
85	art	3	25.5
95	bob	3	63.5
96	frodo	3	25.5

Answer relation:

rating	minage
3	25.5
7	35.0
8	25.5

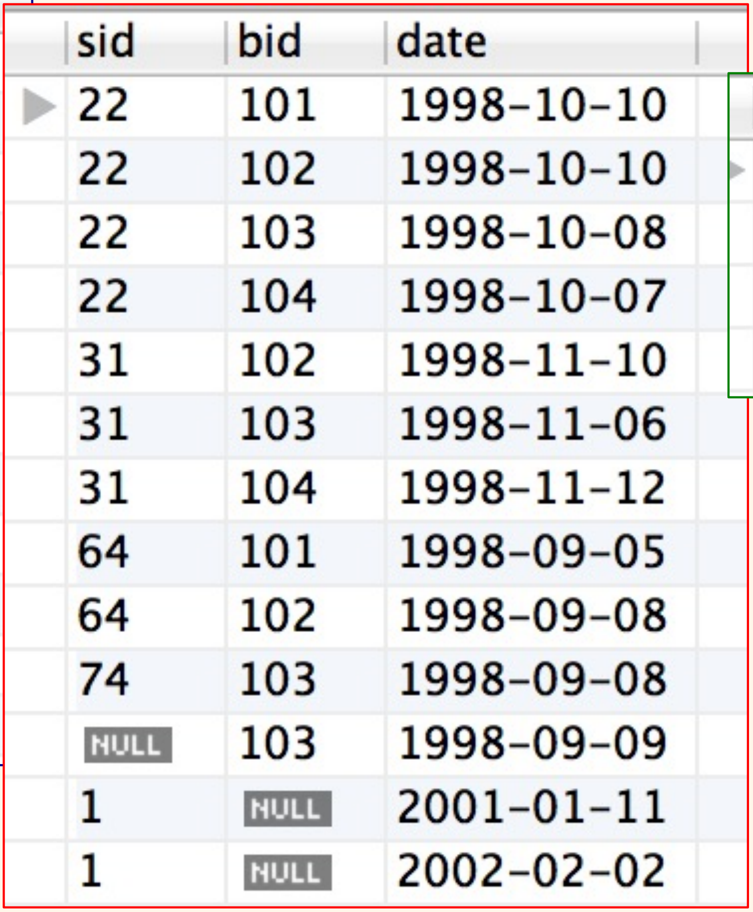
Example Data in MySQL

Sailors



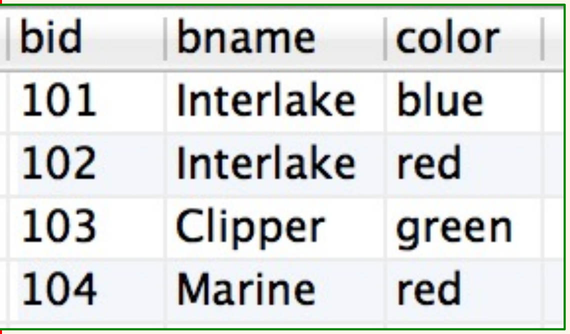
sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	4	25.5
95	Bob	3	63.5
101	Joan	3	NULL
107	Johan...	NULL	35.0

Reserves



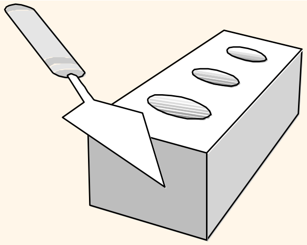
sid	bid	date
22	101	1998-10-10
22	102	1998-10-10
22	103	1998-10-08
22	104	1998-10-07
31	102	1998-11-10
31	103	1998-11-06
31	104	1998-11-12
64	101	1998-09-05
64	102	1998-09-08
74	103	1998-09-08
NULL	103	1998-09-09
1	NULL	2001-01-11
1	NULL	2002-02-02

Boats



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

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

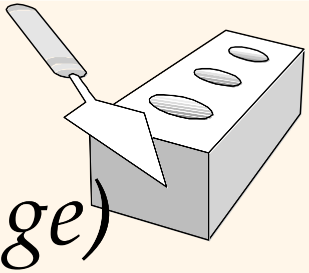


```
SELECT  B.bid, COUNT(*) AS scount
FROM Sailors S, Boats B, Reserves R
WHERE  S.sid=R.sid AND R.bid=B.bid AND B.color= 'red'
GROUP BY B.bid
```

- We're grouping over a join of three relations!
- ***Q:*** What happens if …
  - … we want to get boat names and colors too, or
  - … we remove *B.color= 'red'* from the WHERE clause and add a HAVING clause with this condition?

*(**Hint:** These are "key questions"…☺)*

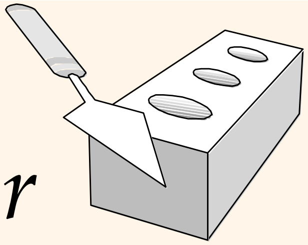
Find age of the youngest sailor with age **>** *18 for each rating with at least 2 sailors (of* ***any*** *age)*



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

- Shows HAVING clause can also contain a subquery.
- Compare this with the query where we considered only ratings with 2 or more sailors *at least 18*!
- What if HAVING clause were replaced by:
  - HAVING COUNT(*) > 1

Find those ratings and average ages for which the average Sailor age is the minimum age over ***all*** *of the Sailors*



- Aggregates themselves can’t be nested! (∴*WRONG…*)

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

- A correct solution (in SQL/92):

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

Compute the average age for *each rating*...

Find the *overall* minimum age

*An Aside: SQL's **WITH** Clause*

***Ex:** Find those ratings and average ages for which the average Sailor age exceeds the average age over all Sailors*

- Our first solution was:

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

- We could use a ***WITH*** clause here for clarity!

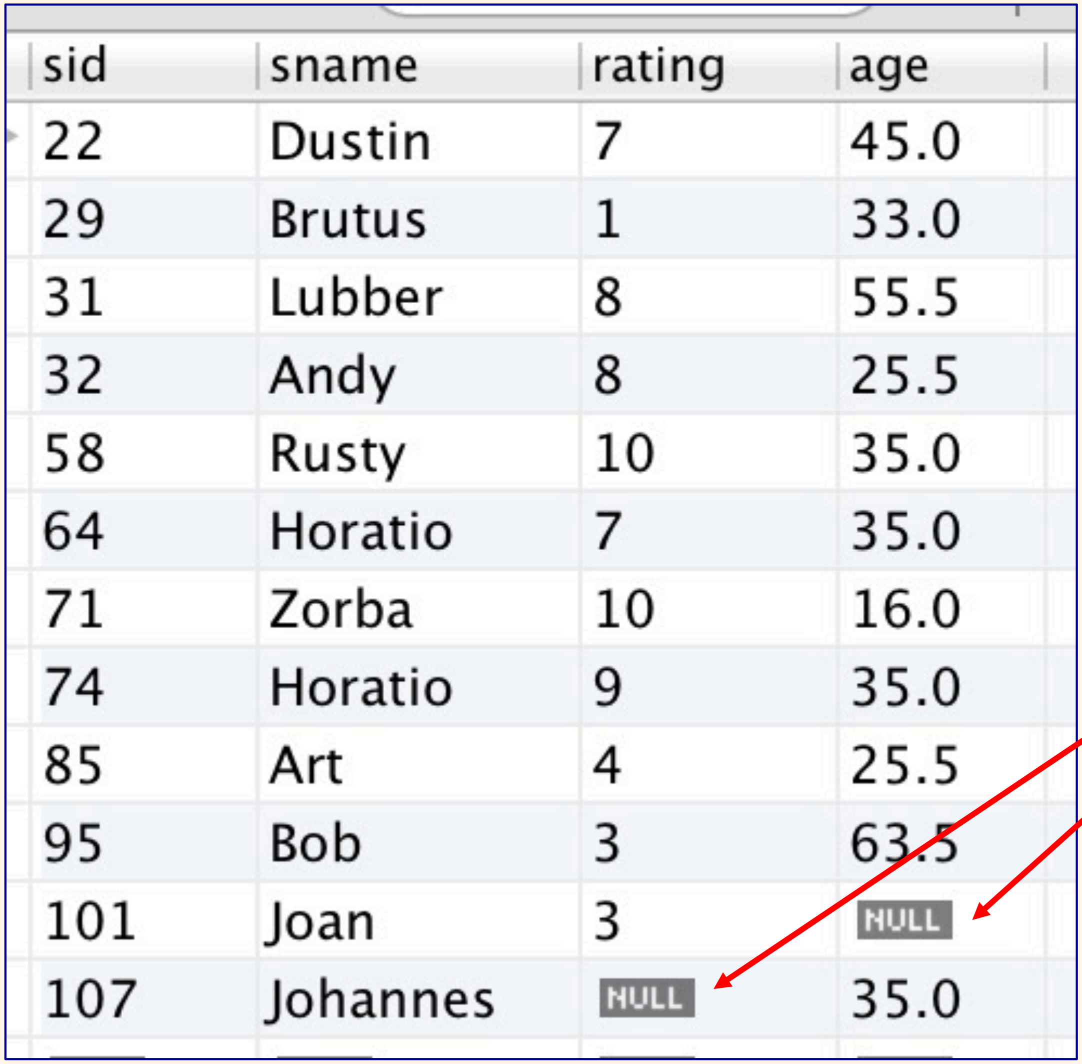
```
WITH Temp AS (SELECT  S.rating, AVG(S.age) AS avgage
                   FROM  Sailors S
                   GROUP BY S.rating)
SELECT  Temp.rating, Temp.avgage        -- (or:  SELECT *)
FROM Temp
WHERE  Temp.avgage > (SELECT AVG(age) FROM Sailors)
```

(Note: slightly different connective operator here...)

Null Values in SQL

- Field values in a tuple are sometimes *unknown* (e.g., a rating has not yet been assigned) or *inapplicable* (e.g., there is no spouse’s name).
  - SQL provides the special value *null* for such situations.
- The presence of *null* complicates many issues. E.g.:
  - Special operators needed to check if value is/is not *null*.
  - Is *rating>8* true or false when *rating* is equal to *null*? What about AND, OR and NOT connectives?
  - We need a 3-valued logic (true, false and *unknown*).
  - Meaning of constructs must be defined carefully. (The WHERE clause eliminates rows that don’t evaluate to ***true***.)
  - New operators (in particular, *outer joins*) possible/needed.

Ex: Sailors w/Some Null Values



sid	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	4	25.5
95	Bob	3	63.5
101	Joan	3	NULL
107	Johannes	NULL	35.0

Q: Which kind(s) of *null* are each of these null values?

Nulls and SQL's 3-Valued Logic

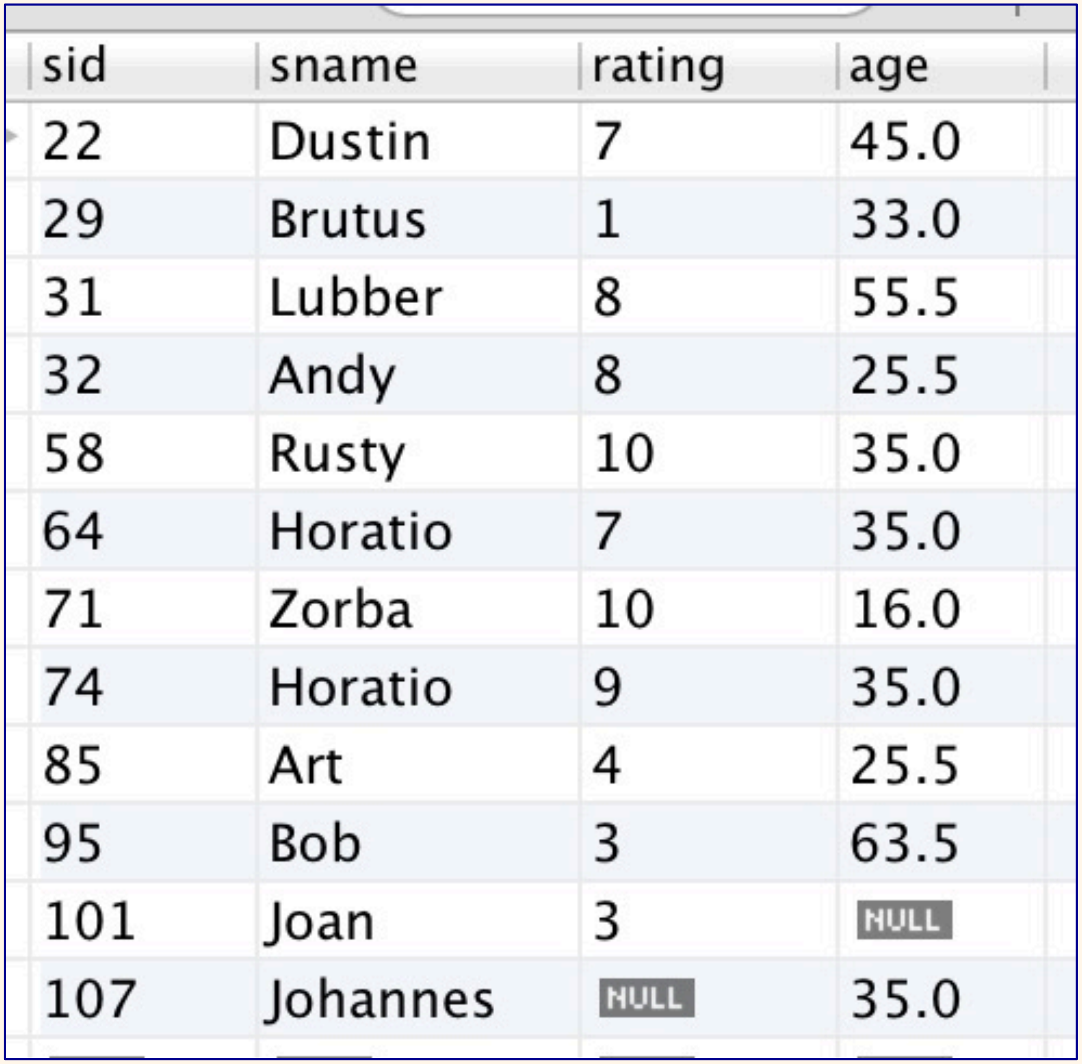
AND	**true**	**false**	**unknown**
true	true	false	unknown
false	false	false	false
unknown	unknown	false	unknown

OR	**true**	**false**	**unknown**
true	true	true	true
false	true	false	unknown
unknown	true	unknown	unknown

NOT	
true	false
false	true
unknown	unknown

Note: SQL arithmetic expressions involving *null* values will yield *null* values (*Ex*: EMP.sal + EMP.bonus)

Basic SQL Queries w/Nulls



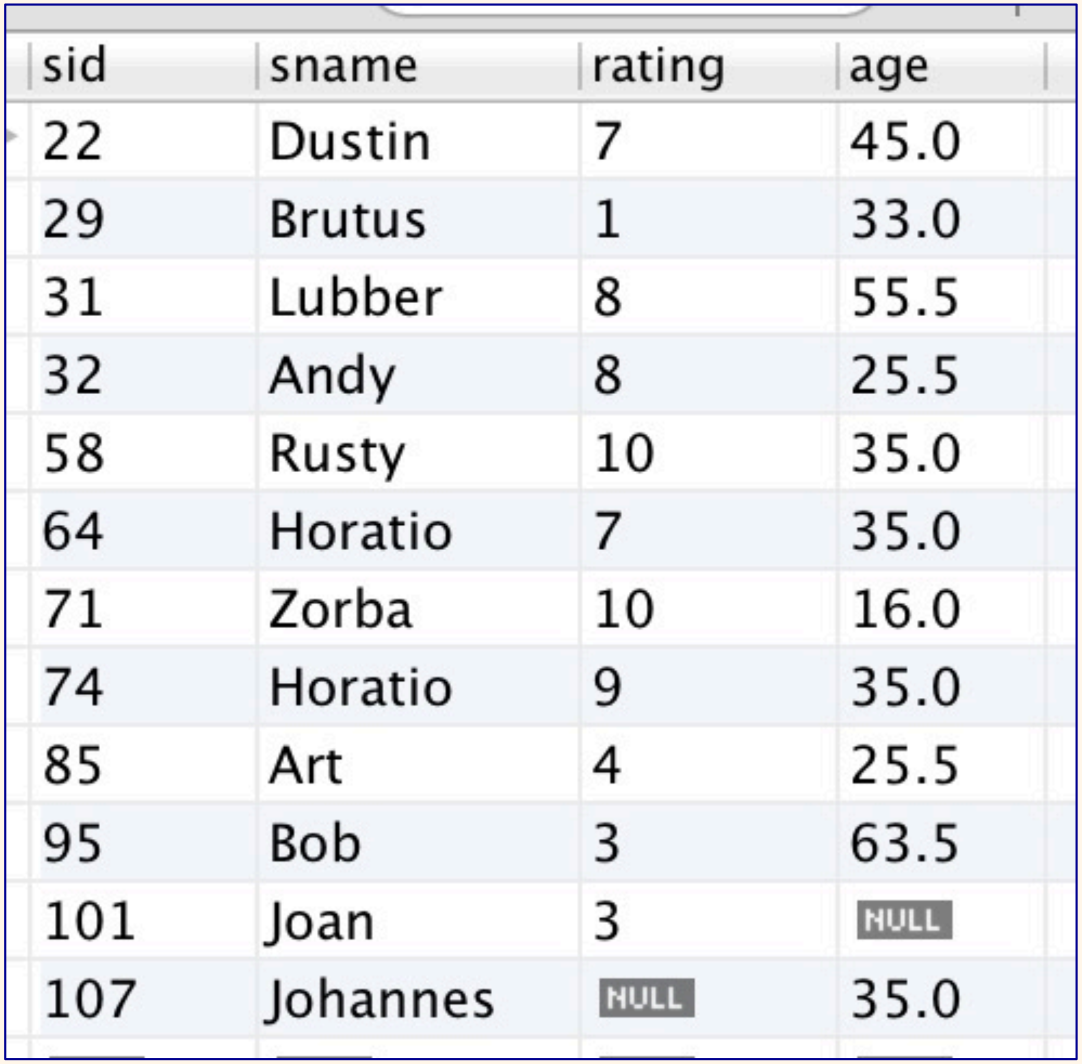
sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	4	25.5
95	Bob	3	63.5
101	Joan	3	NULL
107	Johannes	NULL	35.0

```
SELECT  *
FROM  Sailors S
WHERE age > 35.0
```

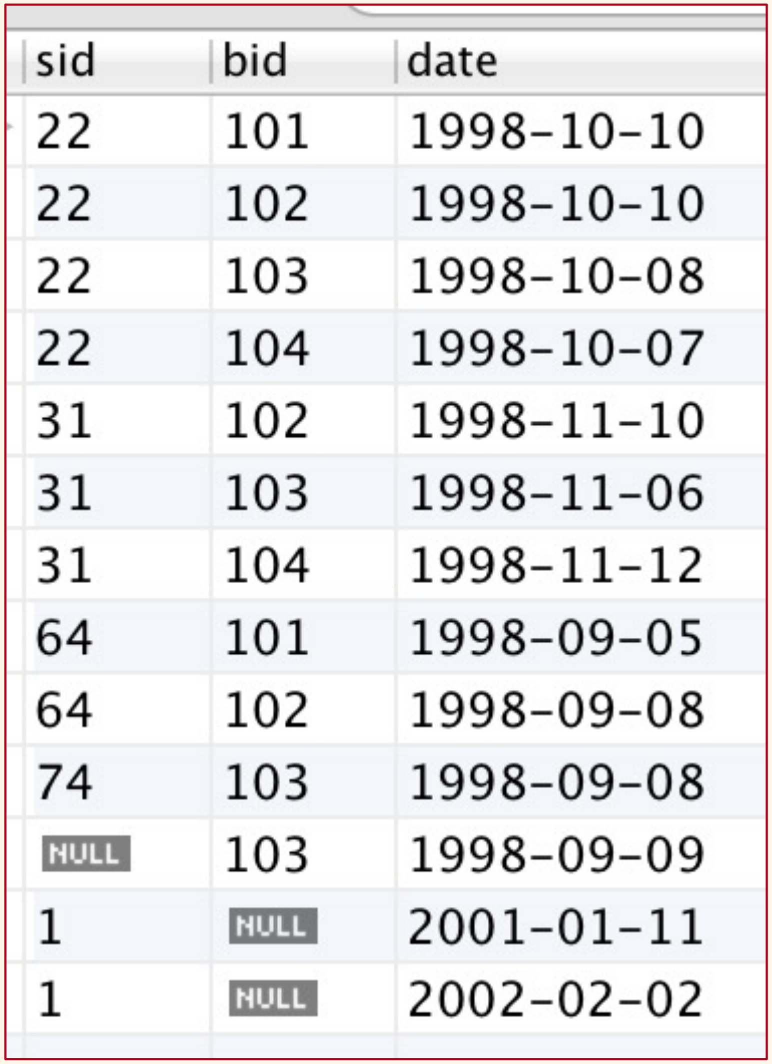
```
SELECT  *
FROM  Sailors S
WHERE age <= 35.0
```

```
SELECT COUNT(*)
FROM  Sailors S
WHERE age > 35.0
        OR age <= 35.0
        OR age IS NULL
```

*Ex: Sailors **and Reserves** w/Nulls*



sid	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	4	25.5
95	Bob	3	63.5
101	Joan	3	NULL
107	Johannes	NULL	35.0



sid	bid	date
22	101	1998-10-10
22	102	1998-10-10
22	103	1998-10-08
22	104	1998-10-07
31	102	1998-11-10
31	103	1998-11-06
31	104	1998-11-12
64	101	1998-09-05
64	102	1998-09-08
74	103	1998-09-08
NULL	103	1998-09-09
1	NULL	2001-01-11
1	NULL	2002-02-02

To Be Continued...