SQL Queries

Chapter 5

https://lagunita.stanford.edu/courses/DB/SQL/SelfPaced/about

Structured Query Language

• Create a Table	`Create (Delete/Modify)`
• Add new records	`Insert`
• Retrieve records	`Select`
• Update records	`Update`
• Delete records	`Delete`
• Create a View	`Create`
• Update a View	`Update`

DDL

SQL Query Language

- Implements relational algebra…

  Select, Project, Join, Set operators

“Relationally complete”

- And so much more…
  - Correlated subqueries
  - Ordering of results
  - Aggregate queries (e.g., SUM, MAX, AVG)
  - Three-valued logic for NULL values
  - Etc.

Basic SQL Query

Optional

Attributes from input relations

```
SELECT [DISTINCT] attr-list
FROM relation-list
WHERE qualification
```

List of relations

Attr1 op Attr2
OPS: <, >, =, <=, >=, <>
Combine using
AND, OR, NOT

(Conceptual) Evaluation:

1. Take cross-product of relation-list
2. Select rows satisfying qualification
3. Project columns in attr-list
   (eliminate duplicates only if DISTINCT)

Optimizer chooses efficient plan!

Example of Basic Query

- Schema:

```
Sailors (sid, sname, rating, age)
Boats (bid, bname, color)
Reserves (sid, bid, rday)
```

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	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

- Find the names of sailors who have reserved boat #103

$$\pi_{sname}((\sigma_{bid=103}\text{Reserves}) \bowtie Sailors)$$

```
SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid = R.sid AND R.bid = 103;
```

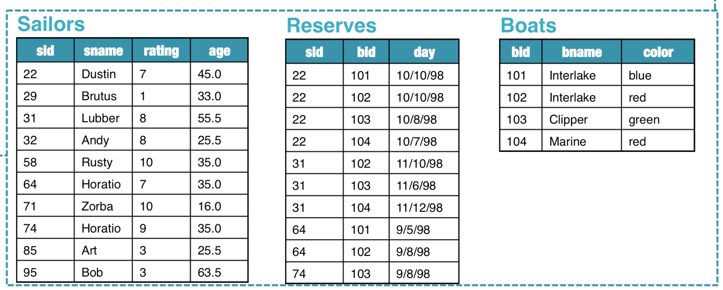
Eliminating Duplicates

```
SELECT DISTINCT sname
FROM Sailors S, Reserves R
WHERE S.sid = R.sid;
```

Another Example

- Schema:

```
Sailors (sid, sname, rating, age)
Boats (bid, bname, color)
Reserves (sid, bid, rday)
```



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	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

- Find the colors of boats reserved by a sailor named Rusty

```
SELECT B.color
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND
      S.sname = 'Rusty';
```

Note on Range Variables

Needed when same relation appears twice in FROM clause

```
SELECT S1.sname, S2.sname
FROM Sailors S1, Sailors S2
WHERE S1.age > S2.age;
```

What does this Query compute?

Good style to always use range variables anyway…

Another Example

- Find pairs of sailors where the first one has half the rating of the second one:

```
SELECT S1.sname AS name1, S2.sname AS name2
FROM Sailors S1, Sailors S2
WHERE 2*S1.rating = S2.rating;
```

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	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

Incrementing the result

- Compute increments for the ratings of persons who have sailed two different boats on the same day:

```
SELECT S.sname, S.rating+1 AS rating
FROM Sailors S, Reserves R1, Reserves R2
WHERE S.sid=R1.sid AND S.sid=R2.sid AND
      R1.day=R2.day AND R1.bid<>R2.bid;
```

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	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

Showing JOINs Explicitly

Find the names of sailors who have reserved boat #103.
All the following are equivalent:

$$\pi_{sname}(\sigma_{bid=103}(Reserves \bowtie Sailors))$$

Cross-product syntax:

```
SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid = R.sid AND
R.bid = 103;
```

Join syntax:

```
SELECT S.sname
FROM Sailors S JOIN Reserves
     R ON S.sid = R.sid
WHERE R.bid = 103;
```

Natural join syntax (joins on common attributes):

```
SELECT S.sname
FROM Sailors S NATURAL JOIN Reserves R
WHERE R.bid = 103;
```

INNER Joins

The join we just saw is also called an INNER JOIN

(We will see outer joins shortly)

Join syntax:

```
SELECT S.sname
FROM Sailors S JOIN Reserves R ON S.sid = R.sid
WHERE R.bid = 103;
```

Eqvt. Inner join syntax:

```
SELECT S.sname
FROM Sailors S INNER JOIN Reserves R ON
     S.sid = R.sid
WHERE R.bid = 103;
```

Example of Basic Query:

Cross-product syntax:

```
SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid = R.sid AND
R.bid = 103;
```

Reserves

sid	bid	rday
22	101	10/10
58	103	11/12

Sailors

sid	sname	rating	age
22	Dustin	7	45
58	Rusty	10	35
31	Lubber	8	55

Reserves x Sailors

sid	bid	rday	sid	sname	rating	age
22	101	10/10	22	Dustin	7	45
22	101	10/10	58	Rusty	10	35
22	101	10/10	31	Lubber	8	55
58	103	11/12	22	Dustin	7	45
58	103	11/12	58	Rusty	10	35
58	103	11/12	31	Lubber	8	55

ORDER BY Clause

Helps sort the result for presentation

Attribute(s) in ORDER BY clause (must be) in SELECT list

Find the names and ages of all sailors, in increasing order of age

```
SELECT S.sname, S.age
FROM Sailors S
ORDER BY S.age [ASC]
```

Find the names and ages of all sailors, in decreasing order of age

```
SELECT S.sname, S.age
FROM Sailors S
ORDER BY S.age DESC
```

ORDER BY Clause

```
SELECT S.sname, S.age, S.rating
FROM Sailors S
ORDER BY S.age ASC, S.rating DESC
```

What does this query compute?

Find the names, ages, and rankings of all sailors.

Sort the result in <u>increasing</u> order of age.

If there is a tie, sort those tuples in decreasing order of rating.

Set Operators

- UNION (eliminates duplicates)
- UNION ALL (keeps duplicates)
- INTERSECT
- EXCEPT or MINUS (set difference)

Union Example

Find names of sailors who have reserved a red or a green boat.

Try without UNION

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	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

Union Example

Find names of sailors who have reserved a red or a green boat.

```
SELECT  DISTINCT S.sname
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid
  AND (B.color = 'red' OR B.color = 'green');
```

```
SELECT  S.sname
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'
UNION
SELECT S.sname
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid and R.bid = B.bid AND B.color = 'green';
```

Intersect

Find names of sailors who have reserved a red and a green boat.

```
SELECT  S.sname
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid
  AND (B.color = 'red' AND B.color = 'green');
```

What is wrong with the above query?

Intersect

Find names of sailors who have reserved a red and a green boat.

```
SELECT  S.sname
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid
  AND (B.color = 'red' AND B.color = 'green');
```

What is wrong with the above query?

Intersect

Find names of sailors who have reserved a red and a green boat.

```
SELECT  S.sname
FROM Sailors S, Reserves R1, Boats B1,
     Reserves R2, Boats B2
WHERE S.sid = R1.sid AND R1.bid = B1.bid
  AND S.sid = R2.sid AND R2.bid = B2.bid
  AND B1.color = 'red' AND B2.color = 'green’
```

Bug?

```
SELECT  S.sname
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'
INTERSECT
SELECT S.sname
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid and R.bid = B.bid AND B.color = 'green';
```

Set Difference Example

Find tuples in A that are not in B

```
SELECT * FROM A
MINUS
SELECT * FROM B;
```

Set Difference Example

Find sids of sailors who have reserved red, but not green boats.

```
SELECT  S.sid
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid
      AND B.color = 'red'
EXCEPT
SELECT  S.sid
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid
      AND B.color = ’green'
```

Simpler?

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	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

Set Difference Example

Find sids of sailors who have reserved red, but not green boats.

```
SELECT  R.sid
FROM Reserves R, Boats B
WHERE R.bid = B.bid AND B.color = 'red'
EXCEPT
SELECT  R.sid
FROM Reserves R, Boats B
WHERE R.bid = B.bid AND B.color = ’green'
```

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	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

More Set Comparison Operators

- Set comparisons:
  - *attr* IN R : true if R contains *attr*
  - EXISTS R : true if R is not an empty relation
  - UNIQUE R : true if no duplicates in R
  - You can use NOT with these, e.g. NOT EXISTS

- Also available ANY or ALL: (op is <, ≤, >, ≥, =, ≠)
  - *attr* > ANY R : some element of R satisfies the condition that attr > that element
  - *attr* < ALL R : all elements of R satisfy the condition that attr < element

Nested Queries

Query with another query embedded inside

```
SELECT  S.sname
FROM  Sailors S
WHERE  S.sid IN
      (SELECT  R.sid
       FROM  Reserves R
       WHERE  R.bid=103)
```

What does this query compute?

Conceptual evaluation:

For each row of Sailors, evaluate the subquery over reserves.

To find sailors who have not reserved 103, use NOT IN.

Over-Use of Nesting

- Common error by novice SQL programmers
- Query optimizers not as good at optimizing queries across nesting boundaries
- Try hard first to write non-nested

```
SELECT  DISTINCT S.sname
FROM Sailors S, Reserves R
WHERE S.sid = R.sid AND R.bid = 103;
```

Example

- Q1: What does this query compute?

```
SELECT S.sid
FROM Sailors S
WHERE S.rating > ANY (SELECT S2.rating
                      FROM Sailors S2
                      WHERE S2.name = ‘John’);
```

- Q2: Rewrite the query without using a nested query

Example

- Q1: What does this query compute?

```
SELECT S.sid
FROM Sailors S
WHERE S.rating > ANY (SELECT S2.rating
                     FROM Sailors S2
                     WHERE S2.name = ‘John’);
```

- Q2: Rewrite the query without using a nested query

```
SELECT DISTINCT S.sid
FROM Sailors S, Sailors S2
WHERE S.rating > S2.rating AND S2.name = ‘John’;
```

Example

Find sailors (all their info) whose rating is greater than that of all sailors called Horatio:

```
SELECT  *
FROM  Sailors S
WHERE  S.rating > ALL  (SELECT  S2.rating
                        FROM  Sailors S2
                        WHERE S2.sname='Horatio')
```

Aggregate Operators

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

```
SELECT COUNT
   (DISTINCT S.name)
FROM Sailors S
```

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

single column*

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

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

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

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	3	25.5
95	Bob	3	63.5

Aggregate Query - Example

Find name and age of the oldest sailor(s)

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

✗

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

How many tuples in the result?

Does not work in sqlite3

```
SELECT  S.sname, S.age
FROM  Sailors S
WHERE  S.age >= ALL (SELECT  S2.age
                     FROM  Sailors S2)
```

GROUP BY

Conceptual evaluation

- Partition data into groups according to some criterion
- Evaluate the aggregate for each group

Example:

For each rating level, find the age of the youngest sailor

```
SELECT   MIN (S.age), S.rating
FROM   Sailors S
GROUP BY   S.rating
```

How many tuples in the result?

GROUP BY and HAVING

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

Target-list contains:
1. Attribute names
(subset of grouping-list)
2. Aggregate operations
e.g. min(age)

Conceptual Evaluation:

1. Eliminate tuples that don't satisfy qualification
2. Partition remaining data into groups
3. Eliminate groups according to group-qualification
4. Evaluate aggregate operation(s) for each group

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

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

Sailors

sid	sname	rating	age
22	Dustin	7	45.0
31	Lubber	8	55.5
71	Zorba	10	16.0
64	Horatio	7	35.0
29	Brutus	1	33.0
58	Rusty	10	35.0

rating	age
1	33.0
7	45.5
7	35.0
8	55.5
10	35.0

rating	age
7	35.0

Answer relation

NULL Values in SQL

- NULL represents ‘unknown’ or ‘inapplicable’
- Query evaluation complications
  - Q: Is (rating > 10) true when rating is NULL?
  - A: Condition evaluates to ‘unknown’ (not T or F)
- What about AND, OR connectives?
  - Need 3-valued logic
- WHERE clause eliminates rows that don’t evaluate to true

p	q	p AND q	p OR q
T	T	T	T
T	F	F	T
T	U	U	T
F	T	F	T
F	F	F	F
F	U	F	U
U	T	U	T
U	F	F	U
U	U	U	U

NULL Values Example

Sailors

sid	sname	rating	age
22	Dustin	7	45
58	Rusty	10	NULL
31	Lubber	8	55

```
SELECT sname
FROM sailors
WHERE age > 45 OR age <= 45
```

What does this query return?

NULL Values in Aggregates

NULL values are generally ignored when computing aggregates

Sailors

sid	sname	rating	age
22	Dustin	7	45
58	Rusty	10	NULL
31	Lubber	8	55

```
SELECT AVG(age)
FROM sailors
```

Returns 50!

Reminder

Inner Joins

Find the names of sailors who have reserved boat #103.
All the following are equivalent:

$$\pi_{sname}(\sigma_{bid=103}(Reserves \bowtie Sailors))$$

Cross-product syntax:

```
SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid = R.sid AND
R.bid = 103;
```

Join syntax:

```
SELECT S.sname
FROM Sailors S JOIN Reserves
     R ON S.sid = R.sid
WHERE R.bid = 103;
```

Natural join syntax (joins on common attributes):

```
SELECT S.sname
FROM Sailors S NATURAL JOIN Reserves R
WHERE R.bid = 103;
```

Outer Joins

Sailors

sid	sname	rating	age
22	dustin	7	45.0
58	rusty	10	35.0

Reserves

sid	bid	day
22	101	10/10/99

```
SELECT S.sid, R.bid
FROM Sailors S NATURAL LEFT[OUTER]
     JOIN Reserves R
```

Result

sid	bid
22	101
58	null

Similarly:

- Right Outer Join
- Full Outer Join

Note: OUTER is default, when using LEFT, RIGHT, or FULL

More Outer Joins

```
SELECT S.sid, R.bid
FROM Sailors S RIGHT[OUTER] JOIN
   Reserves R ON S.sid=R.sid;
```

Sailors

sid	sname	rating	age
58	Rusty	10	35
31	Lubber	8	55

Reserves

sid	bid	rday
22	101	10/10
58	103	11/12

Result

sid	bid
null	101
58	103

JOIN Syntax with Multiple Tables

Sailors

sid	sname	rating	age
22	dustin	7	45.0
58	rusty	10	35.0

Reserves

sid	bid	day
22	101	10/10/99

```
SELECT S.sname, B.bname
FROM Sailors S JOIN Reserves R ON (S.sid = R.sid)
               JOIN Boats B ON (R.bid = B.bid)
WHERE S.name = 'dustin';
```

Similarly:

- RIGHT [OUTER] JOIN ON…
- FULL [OUTER] JOIN ON…
- LEFT [OUTER] JOIN on…
- NATURAL JOINS (outer and inner)

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

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

```
SELECT B.bid,  COUNT (*) AS scount
FROM   Boats B, Reserves R
WHERE R.bid=B.bid

GROUP BY B.bid
HAVING B.color = ‘red’
```

Would this work?
Note: one color per bid

Subtle Errors

Find the sid of sailors who have reserved exactly one boat

```
SELECT S1.sid FROM Sailors S1
MINUS
SELECT R1.sid
FROM Reserves R1, Boats B1, Reserves R2, Boats B2
WHERE R1.sid=R2.sid AND R1.bid=B1.bid
      AND R2.bid=B2.bid AND R1.bid <> R2.bid;
```

There is a subtle error in the above

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	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

Error Fixed

Find the sid of sailors who have reserved exactly one boat

```
SELECT R3.sid FROM Reserves R3
MINUS
SELECT R1.sid
FROM Reserves R1, Boats B1, Reserves R2, Boats B2
WHERE R1.sid=R2.sid AND R1.bid=B1.bid
      AND R2.bid=B2.bid AND R1.bid <> R2.bid;
```

Error Fixed: Another Solution

Find the sid of sailors who have reserved exactly one boat

```
SELECT S.sid
FROM Sailors S, Boats B, Reserves R
WHERE S.sid = R.sid AND B.bid = R.bid
GROUP BY S.sid
HAVING COUNT(*) = 1;
```

Intersect on Non-Key

Find the names of sailors who have reserved a red and a green boat

```
SELECT  S.sname
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'
INTERSECT
SELECT S.sname
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid and R.bid = B.bid AND B.color ='green'
```

What is wrong with the above query?

Fix it: (1) by using a view and (2) without a view.

Error Fixed

Find the names of sailors who have reserved a red and a green boat

```
CREATE VIEW RedGreenSailors AS
SELECT  S.sid
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'
INTERSECT
SELECT S.sid
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid and R.bid = B.bid AND B.color = 'green';

SELECT S.sname
FROM Sailors S, RedGreenSailors R
WHERE S.sid = R.sid;
DROP VIEW RedGreenSailors;
```

Error Fixed: Another Solution

Find the names of sailors who have
reserved a red and a green boat
Get rid of the VIEW

```
SELECT S.sname
FROM Sailors S,
(SELECT  S.sid
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'
INTERSECT
SELECT S.sid
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid and R.bid = B.bid AND B.color = 'green')
RedGreenSailors
WHERE S.sid = RedGreenSailors.sid;
```

Another sol:
See Q8 in
Ch. 5, p.150

Find the 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) AS MINAGE
FROM  Sailors S
WHERE  S.age > 18
GROUP BY  S.rating
HAVING  1 < (SELECT  COUNT (*) FROM  Sailors S2
             WHERE  S2.rating=S.rating)
```

- Subquery in the HAVING clause
- Compare this with the query where we considered only ratings with 2 sailors over 18!

Find the 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) AS MINAGE
FROM  Sailors S
WHERE  S.age > 18
GROUP BY  S.rating
HAVING  1 < (SELECT  COUNT (*) FROM  Sailors S2
             WHERE  S2.rating=S.rating)
```

- Compare this with the query where we considered only ratings with 2 sailors over 18!

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

Find ratings for which the average age is the minimum of the average age over all ratings*

Aggregate operations cannot be nested!
WRONG: →

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

Correct solution (in SQL/92 – may fail in practice):

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

If above does not work, one solution is to define T as a view

Solution Using Views

```
CREATE VIEW AVG_AGE_BY_RATING AS
SELECT S.rating, AVG(S.age) AS avgage
FROM Sailors S
GROUP BY S.rating;


SELECT T.rating, T.avgage
FROM AVG_AGE_BY_RATING T
WHERE T.avgage= (SELECT MIN(A.avgage)
            FROM AVG_AGE_BY_RATING A);
```