More SQL: Aggregration

Serious people can count: Aggregation

```
SELECT COUNT(*)
FROM   Sailors S

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

SELECT COUNT(DISTINCT S.name)
FROM   Sailors S
WHERE  S.name LIKE ‘D%’
```

COUNT([DISTINCT] A)
SUM([DISTINCT] A)
AVG([DISTINCT] A)
MAX/MIN(A)
STDDEV(A)
CORR(A,B)

PostgreSQL documentation
http://www.postgresql.org/docs/9.4/static/functions-aggregate.html

Syntax: `FUNCTION(`*`expression`*`)`

Compute 1 value from set
Can include math (age * 2 + 5)
Can include `DISTINCT`

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

=

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

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

≠

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

Name and age of oldest sailor(s)

~~SELECT S.name, MAX(S.age)~~
~~FROM Sailors S~~

All SELECT values must be aggregates
(except for GROUP BY)

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

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

Multiple aggregates does work

```
SELECT AVG(S.rating), MAX(S.age)
FROM   Sailors S
```

```
        avg          | max
---------------------+-----
 5.6666666666666667 |  39
(1 row)
```

GROUP BY

```
SELECT count(*)
FROM   Reserves R
```

Total number of reservations

What if want reservations per boat?

May not even know all our boats (depends on data)!

If we did, could write (awkward):

```
for boat in [0…10]
   SELECT count(*)
   FROM   Reserves R
   WHERE  R.bid = <boat>
```

GROUP BY

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

grouping-list: expressions that define groups

set of tuples w/ same value for all attributes in grouping-list

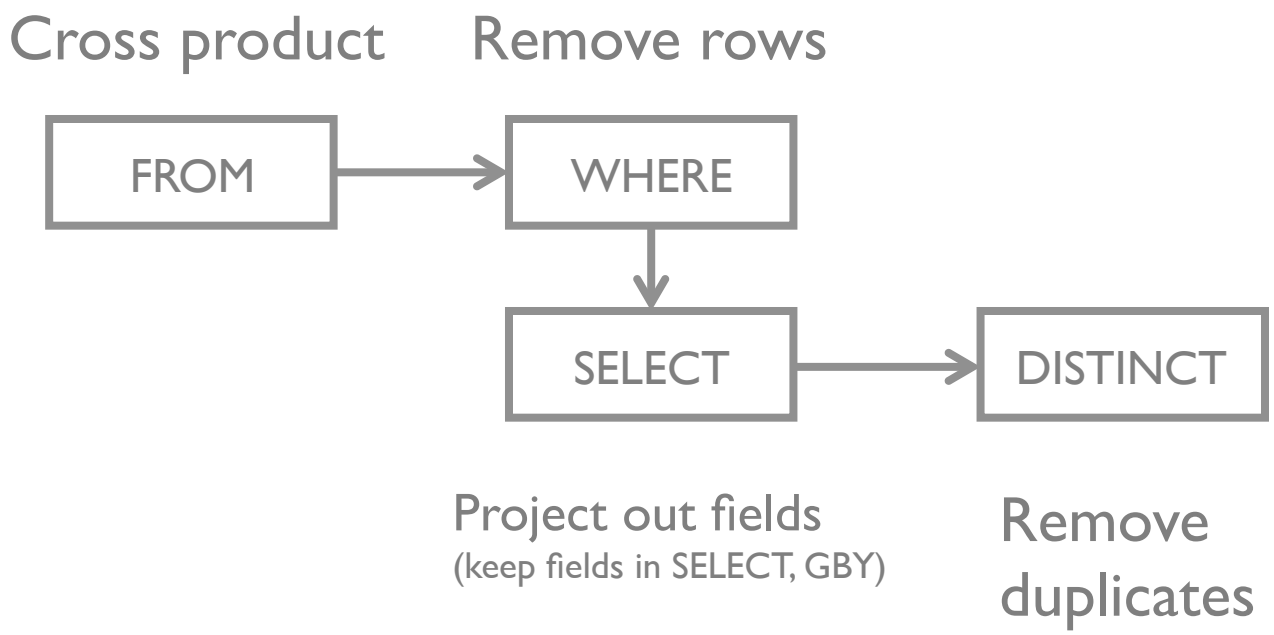
target-list contains

attribute-names $\subseteq$ *grouping-list*

aggregation expressions

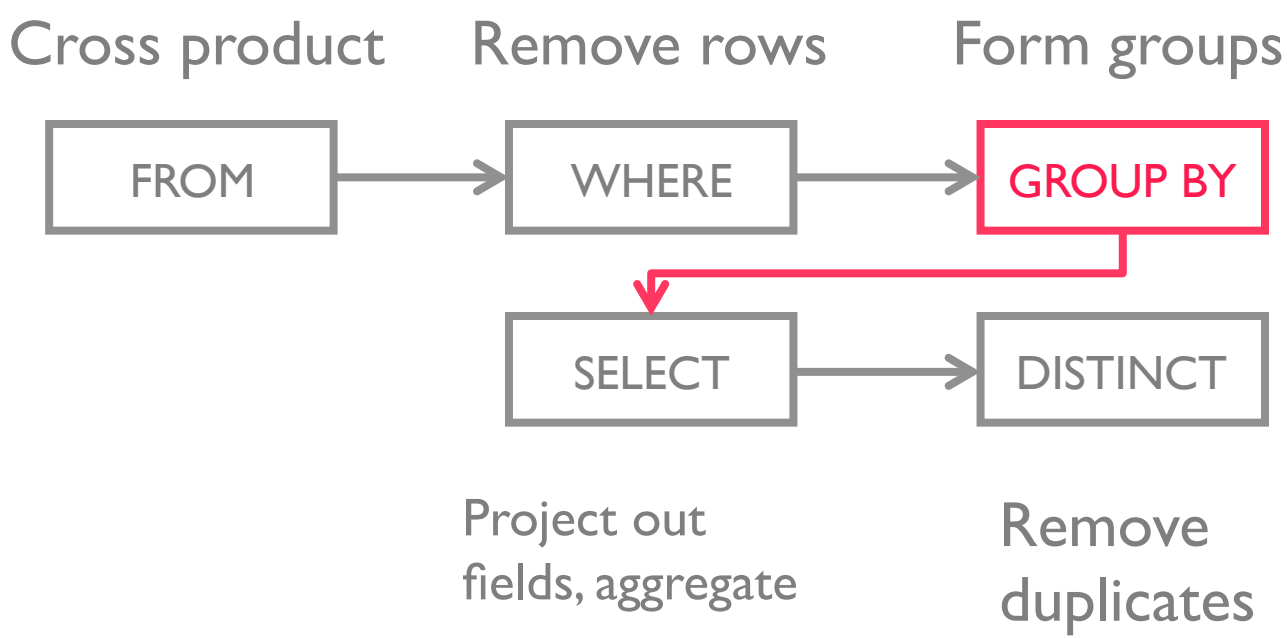
Conceptual Query Evaluation

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



Conceptual Query Evaluation

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



GROUP BY

To get boat ID, name and count:
select r.bid, b.name, count (*)
from reserves r, boats b
where r.bid = b.bid
group by r.bid, b.name

Need to group by b.name as there is no guarantee that b.name is distinct and therefore one output per group

```
SELECT    bid, count(*)
FROM      Reserves R
GROUP BY  bid
```

Number of reservations for each boat

```
SELECT    bid, count(*), sid
FROM      Reserves R
GROUP BY  bid
```

Also show an sid that reserved boat?

Expressions must have *one value per group*:

- In *grouping-list*
- aggregation

HAVING

group-qualification used to remove groups

similar to WHERE clause

Expressions must have *one value per group*

```
SELECT    bid, count(*)
FROM      Reserves R
GROUP BY  bid
HAVING    bid > 50
```

GROUP BY with HAVING

```
SELECT    bid, count(*)
FROM      Reserves R
GROUP BY  bid
HAVING    count(*) > 1
```

Reservations for each boat with more than 1 reservation

~~
```
SELECT    bid, count(*)
FROM      Reserves R
GROUP BY  bid
HAVING    sid > 42
```
~~

```
SELECT    bid, count(*)
FROM      Reserves R
WHERE     sid > 42
GROUP BY  bid
```

GROUP BY

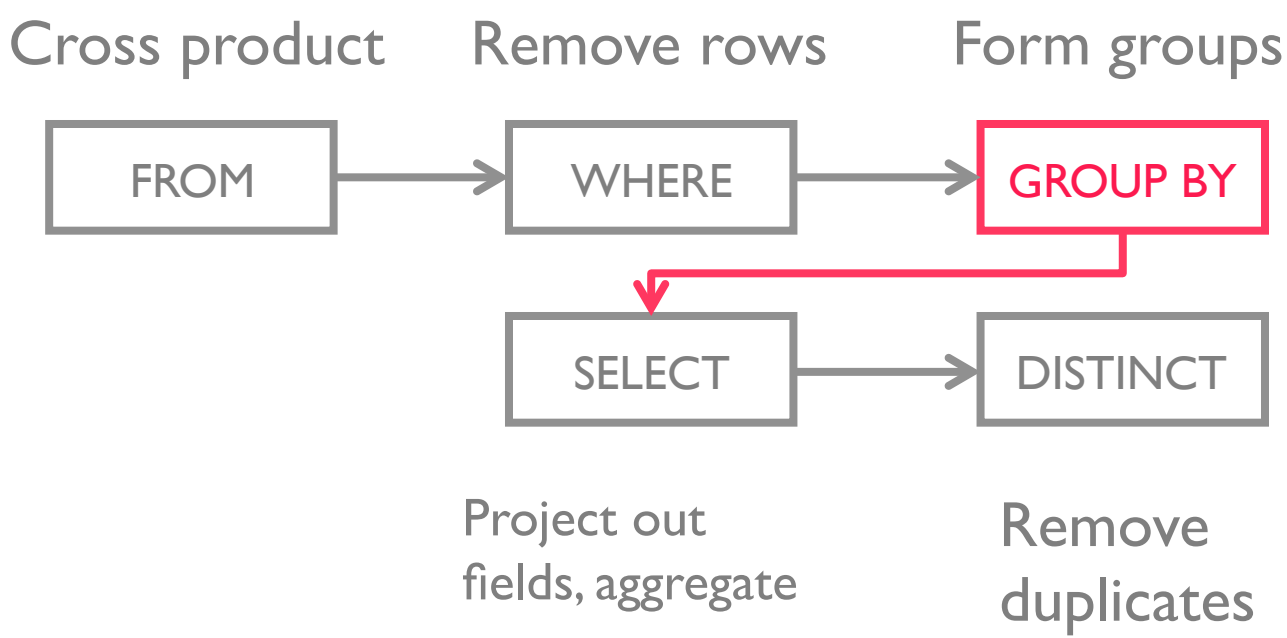
```
SELECT    [DISTINCT] target-list
FROM      relation-list
WHERE     qualification
GROUP BY  grouping-list
HAVING    group-qualification
```

grouping-qualification boolean expression over group

if true, keep the group

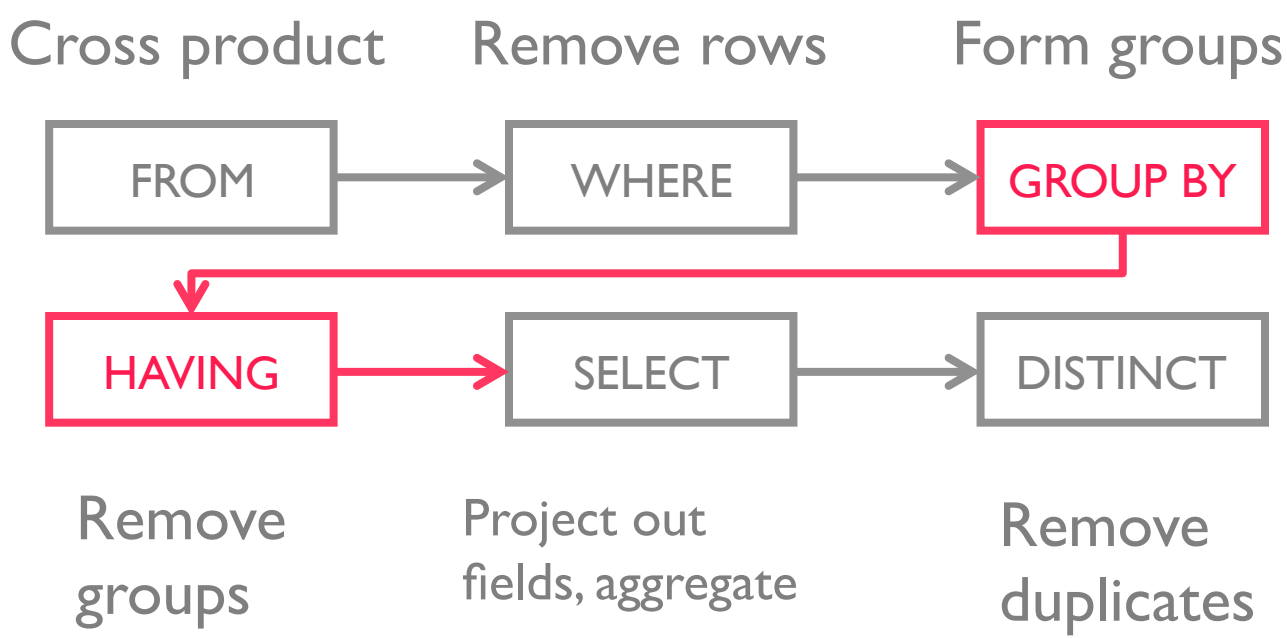
Conceptual Query Evaluation

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



Conceptual Query Evaluation

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



Conceptual Evaluation

sid	bid	day
1	102	9/12
2	102	9/13
4	109	9/20

GROUP BY bid

sid	bid	day
1	102	9/12
2	102	9/13

sid	bid	day
4	109	9/20

HAVING
count(*) > 1

sid	bid	day
1	102	9/12
2	102	9/13

SELECT
bid, count(*)

bid	count
102	2

AVG age of sailors reserving red boats, by rating

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

AVG age of sailors reserving red boats, by rating

```
SELECT    S.rating, S.age
FROM      Sailors S, Boats B, Reserves R
WHERE     S.sid = R.sid AND
          R.bid = B.bid AND
          B.color = ‘red’
```

AVG age of sailors reserving red boats, by rating

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

What if move `B.color='red'` to HAVING clause?

Can't have that as items in having must relate to items in group by .e.g s.rating = 5

ORDER BY

```
SELECT    S.name
FROM      Sailors S
ORDER BY  order-list [ASC|DESC]
```

Order-list: expressions to determine precedence

Left to right: if tie, consider next expression

ASC: Ascending (lowest to highest; default)

DESC: Descending (highest to lowest)

ORDER BY

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

Sailors

sid	name	rating	age
1	Eugene	7	22
2	Luis	2	39
3	Ken	7	27

Result

name	rating	age
Luis	2	39
Ken	7	27
Eugene	7	22

ORDER BY

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

Sailors

sid	name	rating	age
1	Eugene	7	22
2	Luis	2	39
3	Ken	7	27

Result

name	rating	age
Luis	2	39
Eugene	7	22
Ken	7	27

LIMIT

```
SELECT    S.name, S.rating, S.age
FROM      Sailors S
ORDER BY  S.rating ASC,
          S.age DESC
LIMIT     2
```

Only the first 2 results

Sailors

sid	name	rating	age
1	Eugene	7	22
2	Luis	2	39
3	Ken	8	27

Result

name	rating	age
Luis	2	39
Ken	7	27

LIMIT

```
SELECT    S.name, (S.rating/2)::int, S.age
FROM      Sailors S
ORDER BY  (S.rating/2)::int ASC,
          S.age DESC
LIMIT     2 OFFSET 1
```

Only the first 2 results

Sailors

sid	name	rating	age
1	Eugene	7	22
2	Luis	2	39
3	Ken	8	27

Result

name	rating	age
Ken	7	27
Eugene	7	22

LIMIT

```
SELECT    S.name, S.rating, S.age
FROM      Sailors S
ORDER BY  S.rating ASC,
          S.age DESC
LIMIT     (SELECT count(*) / 2 FROM Sailors)
```

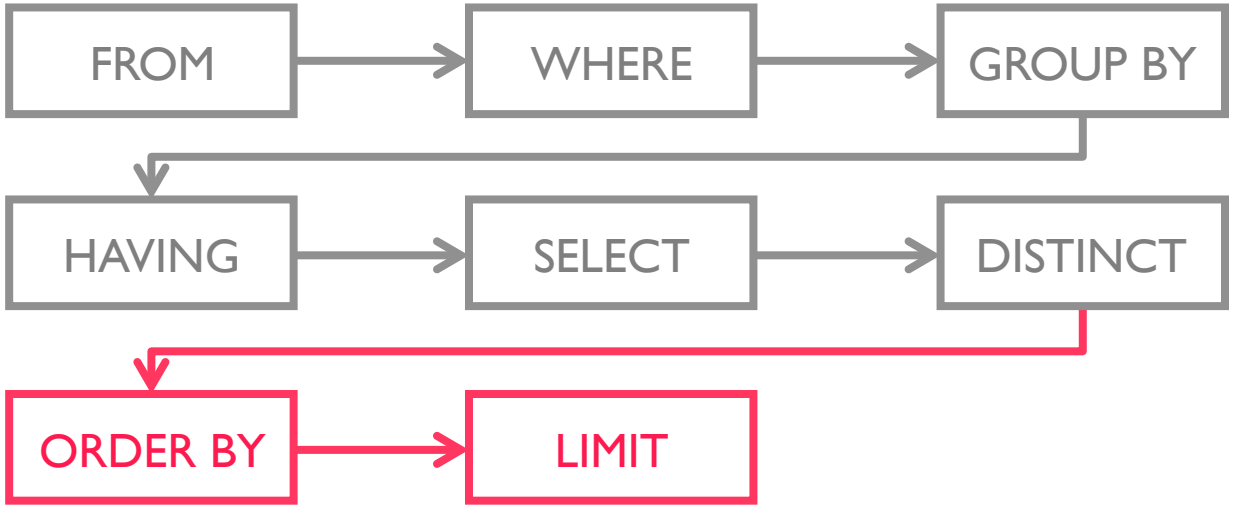
Can have expressions instead of constants

Result

name	rating	age
Luis	2	39

ORDER BY, LIMIT

```
SELECT     [DISTINCT] target-list
FROM       relation-list
WHERE      qualification
GROUP BY   grouping-list
HAVING     group-qualification
ORDER BY   order-list
LIMIT      limit-expr [OFFSET offset-expr]
```



NULL

Field values sometimes unknown or inapplicable

SQL provides a special value *null* for such situations.

The presence of null complicates many issues e.g.,

Is age = null true or false?

Is null = null true or false?

Is null = 8 OR 1 = 1 true or false?

Special syntax "IS NULL" and "IS NOT NULL"

3 Valued Logic (true, false, unknown)

How does WHERE remove rows?

if qualification doesn't evaluate to true

New operators (in particular, outer joins) possible/needed.

NULL

(null > 0) = null

(null + 1) = null

(null = 0) = null

(null AND true) = null

null is null = true

Some truth tables

AND	T	F	NULL
T	T	F	NULL
F	F	F	F
NULL	NULL	F	NULL

OR	T	F	NULL
T	T	T	T
F	T	F	NULL
NULL	T	NULL	NULL