

**Continuing with nested queries.** In last lecture we looked at a query with a subquery nested in the WHERE clause.

Q: Find the names of sailors who have reserved boat #103.

Solution1:

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

Solution2:

```
SELECT S.name
FROM   Sailors S
WHERE  EXISTS (SELECT *
               FROM   Reserves R
               WHERE  R.bid = 103
               AND S.sid = R.sid)
```

Other SQL set operations include:

( ) IN ( )

EXISTS ( )

UNIQUE ( )

NOT can be added in front of IN, EXISTS, or UNIQUE

( ) op ANY ( )

( ) op ALL ( )      op ∈ {<, <=, =, <>, >=, >}

Q: Find the names of sailors with the highest rating.

```
SELECT S.name
FROM   Sailors S
WHERE  EXISTS (SELECT *
               FROM   Reserves R
               WHERE  R.bid = 103
               AND S.sid = R.sid)
```

### Division

Q: Find the names of sailors who have reserved **ALL** boats.

**Solution1:** reword the query: Find the names of sailors such that there is no boat B in the boat table that is not associated with Sailors in a reservation.

```
SELECT S.name
FROM   Sailors S
WHERE  NOT EXISTS [SELECT B.bid
                  FROM   Boats B, Reserves R
                  WHERE  NOT EXIST (SELECT R.bid
                                    FROM   Reserves R
                                    WHERE  R.bid = B.bid
                                    AND R.sid = S.sid) ]
```

Solution2: Algebraic Solution - Division

R	
X	Y
x1	y1
x2	y2
...	...
xi	yi
...	ym

T
Y
y1
y2
...
yn



Set1: All tuples in T: { y1, y2,...yn}  
 Set2: tuples in R2 that are associated with xi in R  
 For each xi  
     if(Set1 - Set2 == NULL)  
     Then xi qualifies.

```
SELECT S.name
FROM   Sailors S
WHERE  NOT EXISTS (Set1 EXCEPT Set2)
```

```
Set1:  SELECT B.bid          Set2:  SELECT R.bid
        FROM   Boats B          FROM   Reserves R
                                     WHERE  R.sid = S.sid
```

**Aggregates** give statistical information that is not in database.

COUNT([DISTINCT A]) - one single column COUNT(\*)  
 SUM([DISTINCT A])  
 AVG([DISTINCT A])  
 MAX(A) - DISTINCT does not make a difference  
 MIN(A) - DISTINCT does not make a difference

Ex. SELECT COUNT(\*) -returns the number of tuples in Sailors.  
 FROM Sailors

SELECT COUNT(\*) -returns N x M. Where N is tuples in Sailors  
 FROM Sailors, Reserves and M is tuples in Reserves.

Q: Find name of sailors with highest rating.

```
SELECT S.name
FROM   Sailors S
WHERE  S.rating IN (SELECT MAX(S1.rating)
                   FROM Sailors S1)
```

If aggregates are used in the SELECT clause, exactly one tuple will be returned!

```
SELECT MAX(rating) -returns 1 tuple with 2 attributes
       AVG(rating)
```

```
SELECT S.sname, Max(S.rating) **WRONG**
FROM   Sailors S
```

SELECT must be just for aggregates. The example above does not make sense, there may be more than one snames associated with the maximum rating.

**GROUP BY**

Syntax:

```
SELECT  [DISTINCT] target_list
FROM    relation_list
WHERE   qualifications
GROUP BY grouping_list
Having group_qualifications
```

**Evaluation - Originally**

1. cross-product
2. select tuples based on *qualifications*
3. project the columns needed

**Evaluation - with Group By and (optional) Having**

1. cross-product
2. select tuples based on *qualifications*
3. Group tuples based on *grouping.list*
4. Ignore those groups where *group\_qualification* returns FALSE
5. project the columns needed