**Nested Queries**
Sailors(sid, sname, rating, age)
Reserve(sid, bid, day)
Boats(bid, bname, color)

The names of sailors that reserved boat 103
SELECT S.sname
FROM Sailors S
WHERE S.sid IN (SELECT R.sid
    FROM Reserve R
    WHERE R.bid = 103);

SELECT S.sname
FROM Sailors S
WHERE EXISTS (SELECT *
    FROM Reserve R
    WHERE R.bid = 103);

**The nested query in this query is a correlated subquery.**

IN => “is a member of set”
EXISTS => “is size of set at least one?”

Find sailors with the highest rating.
- Cannot use max(...) in WHERE clause without running a query; have to use a subquery.
SELECT S.name
FROM Sailors S
WHERE S.rating = (SELECT S2.max(rating)
    FROM Sailor S2);

SELECT S.name
FROM Sailors S
WHERE S.rating >= ALL (SELECT S2.rating
    FROM Sailors S2);

**Exercises**
i) List sailors’ sids whose rating is higher than at least one of the sailors whose name is Horatio.

SELECT S.sid
FROM Sailors S
WHERE S.rating > ANY (SELECT S2.rating
    FROM Sailor S2
    WHERE S2.name = ‘Horatio’);
ii) List sailors’ sids whose rating is higher than all sailors named Horatio.

```
SELECT S.sid
FROM Sailor S
WHERE S.rating > ALL (SELECT S2.rating
    FROM Sailor S2
    WHERE S2.name = ‘Horatio’);
```

-These cannot be expressed with a simple SELECT-FROM-WHERE.
  Proof: 1) Observe all SELECT-FROM-WHERE queries are monatomic.
    -Monatomic queries are queries whose results can only stay the same or grow (not shrink) as databases grow.
    2) This query is non-monatomic, thus cannot be expressed with SELECT-FROM-WHERE.

We have been using existential conditions. (easy)
Now let’s try using universal conditions. (hard)

1) Find boats not reserved by the sailor with sid = 100.
B: all boats
R: boats reserved by sid = 100
-given these sets, the query is B-R

```
SELECT B.bid
FROM Boats B
EXCEPT SELECT B.bid
FROM Reserve R
WHERE R.sid = 100;
```

2) Names of sailors who reserved all boats.
SELECT S.name
FROM Sailor S
WHERE NOT EXISTS (SELECT B.bid
    FROM Boats B
    EXCEPT SELECT R.bid
    FROM Reserves R
    WHERE R.sid = S.sid);

**Nulls and Semi-joins**
Nulls can go into any value, regardless of type.
Meaning of null depends on application.
  -value not available, unknown, etc

For example...

<table>
<thead>
<tr>
<th>name</th>
<th>age</th>
<th>height</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe</td>
<td>20</td>
<td>NULL</td>
<td>200</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
SELECT *
FROM Person P
WHERE P.age > 25 AND (P.height > 6 OR weight < 190)

In SQL, there are three boolean values: true, unknown, and false. Rows are filtered if the WHERE clause evaluates to false or unknown.