SQL: Structured Query Language

Part 2

Chapter 5

Nested Queries: IN

IN: Allows us to test whether a value is in a given set of elements (usually generated by another SQL query)

Find names of sailors who've ids 1 or 2 or 3 or 4 or 5

```
SELECT S.sname
FROM   Sailors S
WHERE  S.sid IN (1, 2, 3, 4, 5)
```

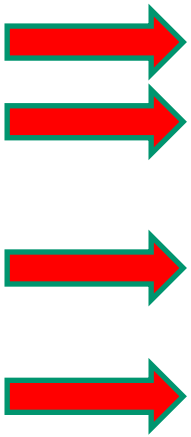
sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
9	Mike	4	57
4	Mary	1	17
22	Jake	10	57
3	Nancy	8	27

Nested Queries: IN

IN: Allows us to test whether a value is in a given set of elements (usually generated by another SQL query)

Find names of sailors who've ids 1 or 2 or 3 or 4 or 5

```
SELECT S.sname
FROM   Sailors S
WHERE  S.sid IN (1, 2, 3, 4, 5)
```



sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
9	Mike	4	57
4	Mary	1	17
22	Jake	10	57
3	Nancy	8	27

sname
Fred
Jim
Mary
Nancy

Nested Queries: IN

Find names of sailors who've reserved boat #102:

```
SELECT S.sname
FROM   Sailors S
WHERE  S.sid IN
                (SELECT R.sid
                 FROM   Reserves R
                 WHERE  R.bid=102)
```

sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
3	Nancy	8	27

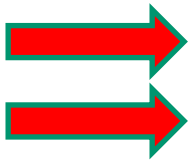
sid	bid	day
1	102	9/12
2	102	9/13
2	101	10/20
3	104	11/20

Nested Queries: IN

Find names of sailors who've reserved boat #102:

```
SELECT S.sname
FROM   Sailors S
WHERE  S.sid IN
                (SELECT R.sid
                 FROM   Reserves R
                 WHERE  R.bid=102)
```

sid
1
2



sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
3	Nancy	8	27

sid	bid	day
1	102	9/12
2	102	9/13
2	101	10/20
3	104	11/20

Find sid's of sailors who've reserved a red and a green boat

Reserves

sid	bid	day
1	103	12/9/2015
2	102	13/9/2015
2	103	1/1/2020
3	101	1/1/2020
3	102	5/1/2020

Boats

bid	bname	color
101	Nina	red
102	Pinta	green
103	Santa Maria	blue

Find sid's of sailors who've reserved a red and a green boat

```
(sid of Sailors who reserve red boat
     AND sid IN
          (sid of Sailors who
           reserve green boat)
```

Reserves

sid	bid	day
1	103	12/9/2015
2	102	13/9/2015
2	103	1/1/2020
3	101	1/1/2020
3	102	5/1/2020

Boats

bid	bname	color
101	Nina	red
102	Pinta	green
103	Santa Maria	blue

Find sid's of sailors who've reserved a red and a green boat

```
(sid of Sailors who reserve red boat
     AND sid IN

          (SELECT R2.sid
           FROM Boats B2,Reserves R2
           WHERE R2.bid=B2.bid
                 AND B2.color='green')
```

sid
2
3

Reserves

sid	bid	day
1	103	12/9/2015
2	102	13/9/2015
2	103	1/1/2020
3	101	1/1/2020
3	102	5/1/2020

Boats

bid	bname	color
101	Nina	red
102	Pinta	green
103	Santa Maria	blue

Find sid's of sailors who've reserved a red and a green boat

```
SELECT R.sid
FROM   Boats B, Reserves R
WHERE  R.bid=B.bid
       AND B.color='red'
       AND R.sid IN
                    (SELECT R2.sid
                     FROM Boats B2,Reserves R2
                     WHERE R2.bid=B2.bid
                           AND B2.color='green')
```

sid
2
3

Reserves

sid	bid	day
1	103	12/9/2015
2	102	13/9/2015
2	103	1/1/2020
3	101	1/1/2020
3	102	5/1/2020

Boats

bid	bname	color
101	Nina	red
102	Pinta	green
103	Santa Maria	blue

Find sid's of sailors who've reserved a red and a green boat

```
SELECT R.sid
FROM   Boats B, Reserves R
WHERE  R.bid=B.bid
       AND B.color='red'
       AND R.sid IN
                   (SELECT R2.sid
                    FROM Boats B2,Reserves R2
                    WHERE R2.bid=B2.bid
                          AND B2.color='green')
```

sid
2
3

Reserves

sid	bid	day
1	103	12/9/2015
2	102	13/9/2015
2	103	1/1/2020
3	101	1/1/2020
3	102	5/1/2020

Boats

bid	bname	color
101	Nina	red
102	Pinta	green
103	Santa Maria	blue

Find sid's of sailors who've reserved a red and a green boat

→ INTERSECT can be **re-written** using IN

```
SELECT R.sid
FROM   Boats B, Reserves R
WHERE  R.bid=B.bid
       AND B.color='red'
       AND R.sid IN (SELECT R2.sid
                     FROM Boats B2,Reserves R2
                     WHERE R2.bid=B2.bid
                           AND B2.color='green')
```

Nested Queries: NOT IN

Find <u>names</u> of sailors who've <u>not</u> reserved boat #102:

sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
3	Nancy	8	27

sid	**bid**	**day**
1	102	9/12
2	102	9/13
2	101	10/20
3	104	11/20

Nested Queries: NOT IN

Find <u>names</u> of sailors who've <u>not</u> reserved boat #102:

```
SELECT   S.sname
FROM     Sailors S
WHERE    S.sid NOT IN
     (SELECT  R.sid
      FROM    Reserves R
      WHERE   R.bid=102)
```

sid
1
2

sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
3	Nancy	8	27

sid	bid	day
1	102	9/12
2	102	9/13
2	101	10/20
3	104	11/20

Nested Queries: NOT IN

Find <u>names</u> of sailors who've <u>not</u> reserved boat #102:

```
SELECT  S.sname
FROM    Sailors S
WHERE   S.sid NOT IN
     (SELECT  R.sid
      FROM    Reserves R
      WHERE  R.bid=102)
```

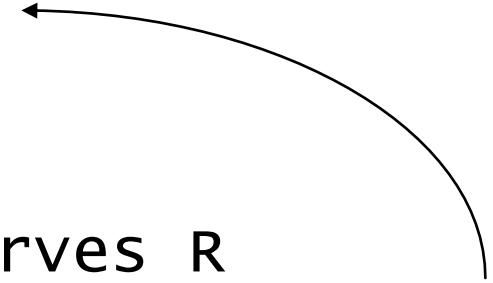
→ EXCEPT can be re-written using NOT IN

Nested Queries with Correlation

EXISTS: Allows us to test whether a set is NON-EMPTY

Find names of sailors who've reserved boat #102:

```
SELECT  S.sname
FROM    Sailors S
WHERE EXISTS
      (SELECT  *
       FROM  Reserves R
       WHERE R.bid=102 AND S.sid=R.sid)
```



- Subquery must be recomputed for each Sailors tuple.
  - Think of subquery as a function call that runs a query

Nested Queries with Correlation

EXISTS: Allows us to test whether a set is NON-EMPTY

Find names of sailors who've reserved boat #102:

sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
3	Nancy	8	27

sid	**bid**	**day**
1	102	9/12
2	102	9/13
2	101	10/20
3	104	11/20

Nested Queries with Correlation

EXISTS: Allows us to test whether a set is NON-EMPTY

Find names of sailors who've reserved boat #102:

```
SELECT  S.sname
FROM    Sailors S
WHERE EXISTS
      (SELECT  *
       FROM  Reserves R
       WHERE R.bid=102 AND S.sid=R.sid)
```

sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
3	Nancy	8	27

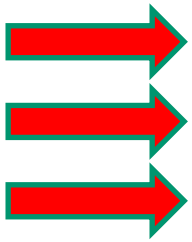
sid	bid	day
1	102	9/12
2	102	9/13
2	101	10/20
3	104	11/20

Nested Queries with Correlation

EXISTS: Allows us to test whether a set is NON-EMPTY

Find names of sailors who've reserved boat #102:

```
SELECT  S.sname
FROM    Sailors S
WHERE EXISTS
      (SELECT  *
       FROM  Reserves R
       WHERE R.bid=102 AND S.sid=R.sid)
```



sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
3	Nancy	8	27

sid	bid	day
1	102	9/12
2	102	9/13
2	101	10/20
3	104	11/20

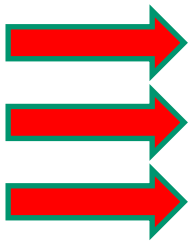
Nested Queries with Correlation

EXISTS: Allows us to test whether a set is NON-EMPTY

Find names of sailors who've reserved boat #102:

```
SELECT  S.sname
FROM    Sailors S
WHERE EXISTS
      (SELECT  *
       FROM  Reserves R
       WHERE R.bid=102 AND S.sid=R.sid)
```

Non Empty 1 Non Empty 2 Empty 3



sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
3	Nancy	8	27

sid	bid	day
1	102	9/12
2	102	9/13
2	101	10/20
3	104	11/20

Nested Queries with Correlation

Find sailors with <u>at most one</u> reservation for boat #102

Nested Queries with Correlation

Find sailors with <u>at most one</u> reservation for boat #102

```
SELECT  S.sname
FROM    Sailors S
WHERE UNIQUE
      (SELECT  R.bid
       FROM  Reserves R
       WHERE R.bid=102 AND S.sid=R.sid)
```

Nested Queries with Correlation

Find sailors with <u>at most one</u> reservation for boat #102

```
SELECT  S.sname
FROM    Sailors S
WHERE UNIQUE
      (SELECT  R.bid
       FROM  Reserves R
       WHERE R.bid=102 AND S.sid=R.sid)
```

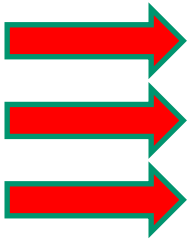
UNIQUE checks for duplicate tuples. When applied to a subquery, it is **TRUE**:

- if no row appers twice,
- if the answer is empty set

Nested Queries with Correlation

Finds sailors with <u>at most one</u> reservation for boat #102

```
SELECT  S.sname
FROM    Sailors S
WHERE UNIQUE
      (SELECT  R.bid
       FROM  Reserves R
       WHERE R.bid=102 AND S.sid=R.sid)
```



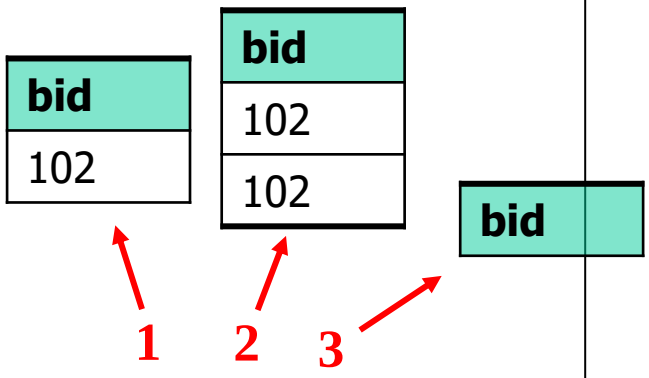
sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
3	Nancy	8	27

sid	bid	day
1	102	9/12
2	102	9/13
2	101	10/20
3	104	11/20
2	102	11/20

Nested Queries with Correlation

Finds sailors with <u>at most one</u> reservation for boat #102

```
SELECT   S.sname
FROM     Sailors S
WHERE UNIQUE
        (SELECT  R.bid
         FROM  Reserves R
         WHERE R.bid=102 AND S.sid=R.sid)
```



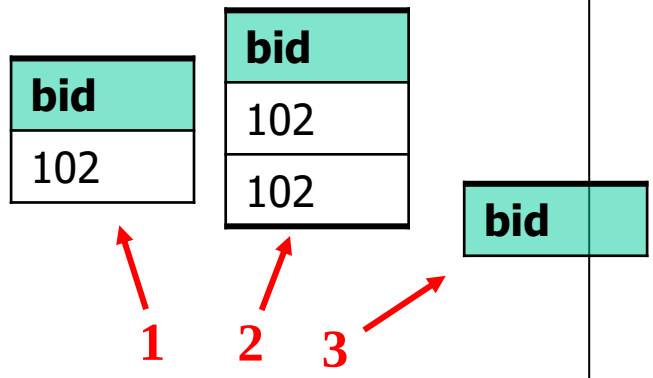
sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
3	Nancy	8	27

sid	bid	day
1	102	9/12
2	102	9/13
2	101	10/20
3	104	11/20
2	102	11/20

Nested Queries with Correlation

Finds sailors with <u>at most one</u> reservation for boat #102

```
SELECT  S.sname
FROM    Sailors S
WHERE UNIQUE
      (SELECT  R.bid
       FROM  Reserves R
       WHERE R.bid=102 AND S.sid=R.sid)
```



Result

sname
Fred
Nancy

sid	sname	rating	age
1	Fred	7	22
2	Jim	2	39
3	Nancy	8	27

sid	bid	day
1	102	9/12
2	102	9/13
2	101	10/20
3	104	11/20
2	102	11/20

In the subquery, what happens **if we write * instead of *R.bid***?

More on Set-Comparison Operators

- We've already seen IN, EXISTS and UNIQUE. Can also use NOT IN, NOT EXISTS and NOT UNIQUE.
- Also available: *op* ANY, *op* ALL $>,<,=,\geq,\leq,\neq$

More on Set-Comparison Operators

- We've already seen IN, EXISTS and UNIQUE. Can also use NOT IN, NOT EXISTS and NOT UNIQUE.
- Also available: *op* ANY, *op* ALL $>,<,=,\geq,\leq,\neq$

Find sailors whose rating is greater than some sailor called Jim:

sid	sname	rating	age
1	Fred	7	20
2	Jim	2	39
9	Mike	7	20
4	Jim	1	17
22	Fred	7	50
3	Nancy	1	21

More on Set-Comparison Operators

- We've already seen IN, EXISTS and UNIQUE. Can also use NOT IN, NOT EXISTS and NOT UNIQUE.
- Also available: *op* ANY, *op* ALL $>,<,=,\geq,\leq,\neq$

Find sailors whose rating is greater than some sailor called Jim:

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

sid	sname	rating	age
1	Fred	7	20
2	Jim	2	39
9	Mike	7	20
4	Jim	1	17
22	Fred	7	50
3	Nancy	1	21

If the subquery returns an **empty set**, comparison returns **FALSE**

More on Set-Comparison Operators

Find sailors whose rating is greater than every sailor called Jim.

sid	sname	rating	age
1	Fred	7	20
2	Jim	2	39
9	Mike	7	20
4	Jim	1	17
22	Fred	7	50
3	Nancy	1	21

More on Set-Comparison Operators

Find sailors whose rating is greater than every sailor called Jim.

sid	sname	rating	age
1	Fred	7	20
2	Jim	2	39
9	Mike	7	20
4	Jim	1	17
22	Fred	7	50
3	Nancy	1	21

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

If the subquery returns an **empty set**, comparison returns **TRUE**

More on Set-Comparison Operators

Find sailors with highest rating.

sid	sname	rating	age
1	Fred	7	20
2	Jim	2	39
9	Mike	7	20
4	Jim	1	17
22	Fred	7	50
3	Nancy	1	21

More on Set-Comparison Operators

Find sailors with highest rating.

sid	sname	rating	age
1	Fred	7	20
2	Jim	2	39
9	Mike	7	20
4	Jim	1	17
22	Fred	7	50
3	Nancy	1	21

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

Note: ***IN*** *equivalent to* ***= ANY***

NOT IN *equivalent to* ***<> ALL***

Division in SQL

Find sailors who've reserved all boats.

$$\rho\ (Tempsids, (\pi_{sid,bid} \text{Reserves}) / (\pi_{bid} Boats))$$

$$\pi_{sname}\ (Tempsids \bowtie Sailors\ )$$

Division in SQL

Find sailors who've reserved all boats.

$$\rho\ (Tempsids, (\pi_{sid,bid} \text{Re}serves) / (\pi_{bid} Boats))$$

$$\pi_{sname}\ (Tempsids \bowtie Sailors\ )$$

```
SELECT  S.sname
FROM  Sailors S
WHERE  NOT EXISTS ( SELECT  B.bid
                    FROM  Boats B
                    WHERE  NOT EXISTS ( SELECT  R.bid
                                        FROM  Reserves R
                                        WHERE  R.bid=B.bid
                                        AND R.sid=S.sid ) )
```

Division in SQL

Find sailors who've reserved all boats.

$$\rho\ (Tempsids, (\pi_{sid,bid} \text{Re}serves) / (\pi_{bid} Boats))$$

$$\pi_{sname}\ (Tempsids \bowtie Sailors\ )$$

```
SELECT  S.sname                          Sailors S such that ...
FROM  Sailors S
WHERE  NOT EXISTS ( SELECT  B.bid
                    FROM  Boats B                  there is no boat B
                    WHERE  NOT EXISTS ( SELECT  R.bid
                                        FROM  Reserves R
                                        WHERE  R.bid=B.bid
                                        AND R.sid=S.sid ) )
```

without a Reserves tuple showing S reserved B

(i.e., there is no reservation tuple)

Division in SQL

Find sailors who've reserved all boats.

```
SELECT  S.sname
FROM  Sailors S
WHERE  NOT EXISTS ( SELECT  B.bid
                    FROM  Boats B
                    WHERE  NOT EXISTS ( SELECT  R.bid
                                        FROM  Reserves R
                                        WHERE  R.bid=B.bid
                                        AND R.sid=S.sid ) )
```

What is the green subquery finding?

Division in SQL

Find sailors who've reserved all boats.

```
SELECT  S.sname
FROM  Sailors S
WHERE  NOT EXISTS ( SELECT  B.bid
                    FROM  Boats B
                    WHERE  NOT EXISTS ( SELECT  R.bid
                                        FROM  Reserves R
                                        WHERE  R.bid=B.bid
                                        AND R.sid=S.sid ) )
```

What is the green subquery finding?

Ans: The boats that are **not reserved** by the given sailor!

sid	name
1	fred
2	wilma

bid	color
101	red
102	green

sid	bid
1	101
1	102
2	102

```
SELECT  S.sname
FROM  Sailors S
WHERE  NOT EXISTS ( SELECT  B.bid
                    FROM  Boats B
                    WHERE  NOT EXISTS ( SELECT  R.bid
                                        FROM  Reserves R
                                        WHERE  R.bid=B.bid
                                        AND R.sid=S.sid ))
```

sid	name
1	fred
2	wilma

bid	color
101	red
102	green

sid	bid
1	101
1	102
2	102

Answer:

sname
fred

```
SELECT  S.sname
FROM  Sailors S
WHERE  NOT EXISTS ( SELECT  B.bid
                    FROM  Boats B
                    WHERE  NOT EXISTS ( SELECT  R.bid
                                        FROM  Reserves R
                                        WHERE  R.bid=B.bid
                                        AND R.sid=S.sid ))
```

Empty → returns True

bid

S.sid=1, R.bid=101 — Non empty

S.sid=1, R.bid=102 — Non empty

sid	name
1	fred
2	wilma

bid	color
101	red
102	green

sid	bid
1	101
1	102
2	102

```
SELECT  S.sname
FROM  Sailors S
WHERE  NOT EXISTS ( SELECT  B.bid
                    FROM  Boats B
                    WHERE  NOT EXISTS ( SELECT  R.bid
                                        FROM  Reserves R
                                        WHERE  R.bid=B.bid
                                        AND R.sid=S.sid ))
```

Non Empty → returns False

bid
101

S.sid=2, R.bid=101 — Empty

S.sid=2, R.bid=102 — Non empty

Answer:

sname
fred

Division using EXCEPT

Find sailors who've reserved all boats.

```
SELECT  S.sname
FROM  Sailors S
WHERE  NOT EXISTS
                ((SELECT  B.bid
                   FROM  Boats B)
                  EXCEPT
                 (SELECT  R.bid
                  FROM  Reserves R
                  WHERE  R.sid=S.sid))
```

sid	name
1	fred
2	wilma

bid	color
101	red
102	green

sid	bid
1	101
1	102
2	102

Division using EXCEPT

Find sailors who've reserved all boats.

```
SELECT  S.sname
FROM  Sailors S
WHERE  NOT EXISTS
                ((SELECT  B.bid
                   FROM  Boats B)
                  EXCEPT
                 (SELECT  R.bid
                  FROM  Reserves R
                  WHERE  R.sid=S.sid))
```

sname
fred

bid
101
102

bid
101
102

sid	name
1	fred
2	wilma

bid	color
101	red
102	green

sid	bid
1	101
1	102
2	102

Division using EXCEPT

Find sailors who've reserved all boats.

```
SELECT  S.sname
FROM  Sailors S
WHERE  NOT EXISTS
                ((SELECT  B.bid
                   FROM  Boats B)
                  EXCEPT
                 (SELECT  R.bid
                  FROM  Reserves R
                  WHERE  R.sid=S.sid))
```

sname
fred

bid
101
102

bid
102

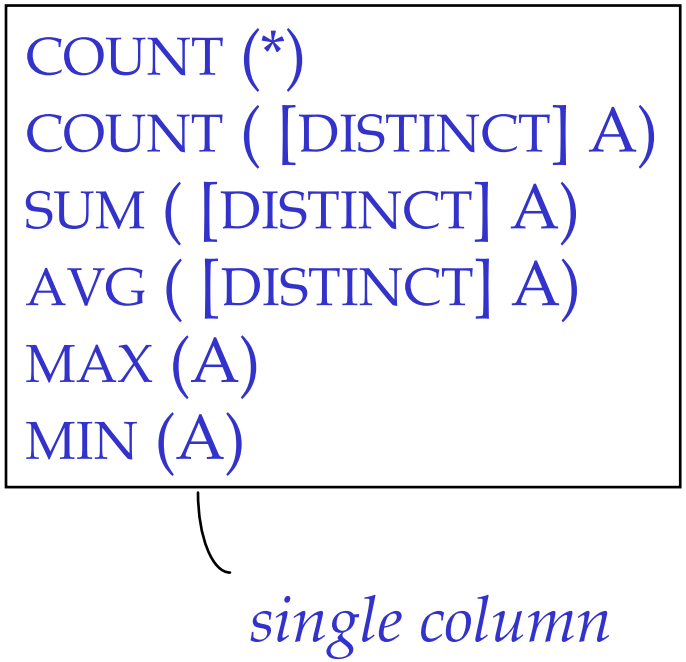
sid	name
1	fred
2	wilma

bid	color
101	red
102	green

sid	bid
1	101
1	102
2	102

Aggregate Operators

- Significant extension of relational algebra.



Aggregate Operators

sid	sname	rating	age
1	Fred	7	20
2	Jim	2	39
9	Mike	7	20
4	Mary	1	17
22	Fred	7	50
3	Nancy	2	21

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

```
SELECT  COUNT (DISTINCT S.rating)
FROM  Sailors S
WHERE S.sname='Fred'
```

```
SELECT  AVG (S.age)
FROM  Sailors S
WHERE  S.rating=7
```

```
SELECT  AVG ( DISTINCT S.age)
FROM  Sailors S
WHERE  S.rating=7
```

Aggregate Operators

Find the names of the sailors with the highest rating.

sid	sname	rating	age
1	Fred	7	20
2	Jim	2	39
9	Mike	7	20
4	Mary	1	17
22	Fred	7	50
3	Nancy	2	21

Aggregate Operators

Find the names of the sailors with the highest rating.

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

sid	sname	rating	age
1	Fred	7	20
2	Jim	2	39
9	Mike	7	20
4	Mary	1	17
22	Fred	7	50
3	Nancy	2	21

Find name and age of the oldest sailor(s)

Find name and age of the oldest sailor(s)

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

Find name and age of the oldest sailor(s)

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

- This query is illegal:
  - If the SELECT clause uses an aggregate operation, then it must use *only* aggregate operations

    (unless the query contains GROUP BY clause)

Find name and age of the oldest sailor(s)

- Correct way to write the query

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

ORDER BY

- Orders the results of a query by the specified fields
- If included in a query, it should be the last part
- Can order by multiple fields by separating the fields with commas, the field that comes before others has precedence in ordering

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

First orders by rating in ascending order, in case of ties between ratings, orders by age in descending order

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

ORDER BY

- Orders the results of a query by the specified fields
- If included in a query, it should be the last part
- Can order by multiple fields by separating the fields with commas, the field that comes before others has precedence in ordering

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

First orders by rating in ascending order, in case of ties between ratings, orders by age in descending order

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	45.0
7	35.0
8	55.5
10	35.0
10	16.0