# SQL – part 2

# SQL – up to now

- The simplest SQL query:
```
SELECT *
FROM Students S
```

- Projection:
```
SELECT S.name, S.login
FROM Students S
```

- Selection:
```
SELECT *
FROM Students S
WHERE S.age=18
```

- Join:
```
SELECT S.name, E.cid
 FROM Students S, Enrolled E
 WHERE S.sid=E.sid AND E.grade='B'
```

# SQL – up to now

- Aggregation:
  - COUNT(*) / COUNT( [DISTINCT] A )
  - SUM( [DISTINCT] A )
  - AVG( [DISTINCT] A )
  - MAX( A )
  - MIN( A )

Examples:

```
SELECT  COUNT (*)
FROM  Sailors S

SELECT  AVG (S.age)
FROM  Sailors S
WHERE
S.rating=10
SELECT  COUNT (DISTINCT
S.rating)
FROM  Sailors S
WHERE S.sname='Bob'
```

# Nested Queries

- Powerful feature of SQL: A SQL query can itself contain a SQL query!
- *Find the name and age of the oldest sailor(s)*

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

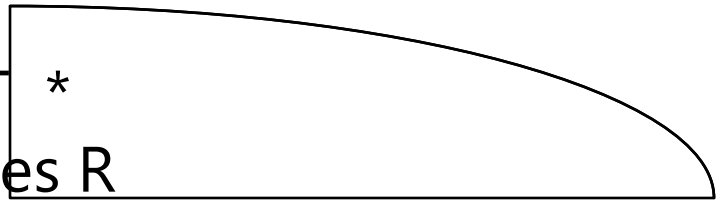
- Useful keywords: IN and EXISTS

# Nested Queries – IN & EXISTS

- *Names of sailors who have reserved boat #103*

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

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



# Nested Queries – IN & EXISTS

- To find sailors who've *not* reserved #103
  - NOT IN or NOT EXISTS
- To understand semantics of IN and EXISTS:
  - think of a *nested loop* evaluation: *For each Sailor tuple, check the qualification by computing the subquery.*

# Expressions

- Can use arithmetic expressions in SELECT clause (plus other operations we'll discuss later)
- Use AS to provide column names

```
SELECT S.age, S.age-5 AS age1, 2*S.age AS age2
FROM  Sailors S
WHERE  S.sname = 'dustin'
```

- Can also have expressions in WHERE clause:

```
SELECT  S1.sname AS name1, S2.sname AS name2
FROM  Sailors S1, Sailors S2
WHERE  2*S1.rating = S2.rating - 1
```

# String operations

- SQL also supports some string operations
- "LIKE" is used for string matching.

```
SELECT  S.age, age1=S.age-5, 2*S.age AS age2
FROM  Sailors S
WHERE  S.sname LIKE 'B_%b'
```

`_' stands for any one character

`%' stands for 0 or more arbitrary characters.

# More Operations

- SQL queries produce new tables
- If the results of two queries are **set-compatible** (same # and types of columns) then we can apply logical operations
  - UNION
  - INTERSECTION
  - SET DIFFERENCE (called EXCEPT or MINUS)

# Find sid's of sailors who've reserved a red **or** a green boat

- UNION: Can be used to compute the union of any two *union-compatible* sets of tuples (which are themselves the result of SQL queries).

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

Vs.

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

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

- If we simply replace OR by AND in the previous query, we get the wrong answer. (Why?)
- Instead, could use a self-join:

```
SELECT R1.sid
FROM Boats B1, Reserves R1,
            Boats B2, Reserves R2
WHERE R1.sid=R2.sid
  AND R1.bid=B1.bid
  AND R2.bid=B2.bid
  AND (B1.color='red' AND B2.color='green')
```

# AND Continued…

- INTERSECT: discussed in the book. Can be used to compute the intersection of any two *union-compatible* sets of tuples.

- Also in text: EXCEPT (sometimes called MINUS)
- Included in the SQL/92 standard, but some systems don't support them.

Key field!

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

# Your turn …

1. Find (the names of) all sailors who are over 50 years old
2. Find (the names of) all boats that have been reserved at least once
3. Find all sailors who have not reserved a red boat (hint: use "EXCEPT")
4. Find all pairs of same-color boats
5. Find all pairs of sailors in which the older sailor has a lower rating

# Answers …

1. Find (the names of) all sailors who are over 50 years old

```
SELECT S.sname
FROM   Sailors S
WHERE  S.age > 50
```

# Answers …

2. Find (the names of) all boats that have been reserved at least once

```
SELECT DISTINCT B.bname
FROM   Boats B, Reserves R
WHERE  R.bid=B.bid
```

# Answers …

3. Find all sailors who have not reserved a red boat

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

# Answers …

4. Find all pairs of same-color boats

```
SELECT B1.bname, B2.bname
FROM   Boats B1, Boats B2
WHERE  B1.color = B2.color
           AND B1.bid < B2.bid
```

# Answers …

5. Find all pairs of sailors in which the older sailor has a lower rating

```
SELECT S1.sname, S2.sname
FROM   Sailors S1, Sailors S2
WHERE  S1.age > S2.age
       AND S1.rating < S2.rating
```

# More on Set-Comparison Operators

- Other operators: ANY, ALL
- Find sailors whose rating is greater than that of some sailor called Horatio:

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

# Rewriting INTERSECT Queries Using IN

*Find sid's of sailors who've reserved both 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')
```

- Similarly, EXCEPT queries can be re-written using NOT IN.
- How would you change this to find *names* (not *sid*'s) of Sailors who've reserved both red and green boats?

# Query #3 revisited …

3. Find all sailors who have not reserved a red boat (this time, without using "EXCEPT")

# Answer …

3. Find all sailors who have not reserved a red boat

```
SELECT S.sid
FROM   Sailors S
WHERE  S.sid NOT IN
            (SELECT R.sid
                  FROM Reserves R, Boats B
                  WHERE R.bid = B.bid
                              AND B.color = ‘red’)
```

# Another correct answer …

3. Find all sailors who have <u>not</u> reserved a red boat

```
SELECT S.sid
FROM   Sailors S
WHERE  NOT EXISTS
            (SELECT *
                  FROM Reserves R, Boats B
                  WHERE R.sid = S.sid
                              AND R.bid = B.bid
                              AND B.color = ‘red’)
```

# Division in SQL

Find sailors who've reserved all boats.

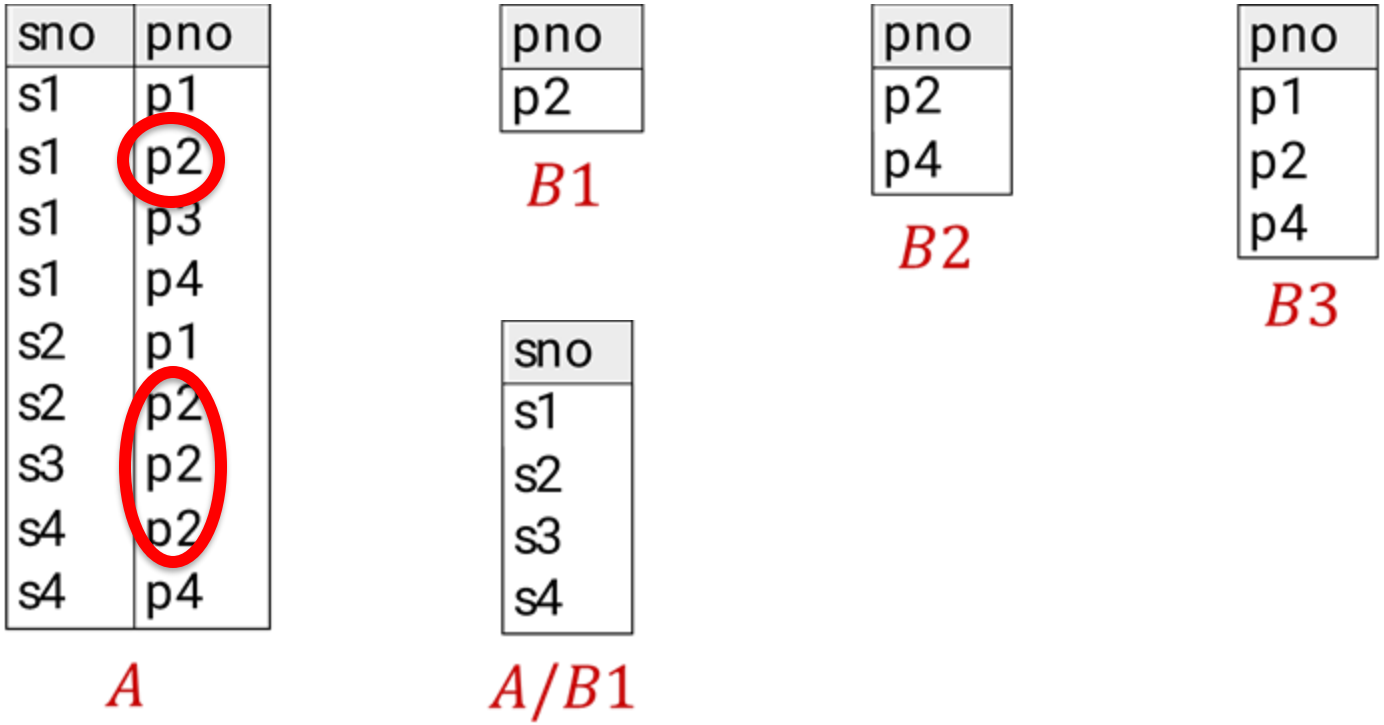
```
SELECT  S.sname
FROM  Sailors S
WHERE  NOT EXISTS  (SELECT  B.bid
                    FROM  Boats B
                    WHERE  NOT EXISTS  (SELECT  R.bid
                                        FROM  Reserves R
                                        WHERE  R.bid=B.bid
                          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

*Sailors S such that ...*

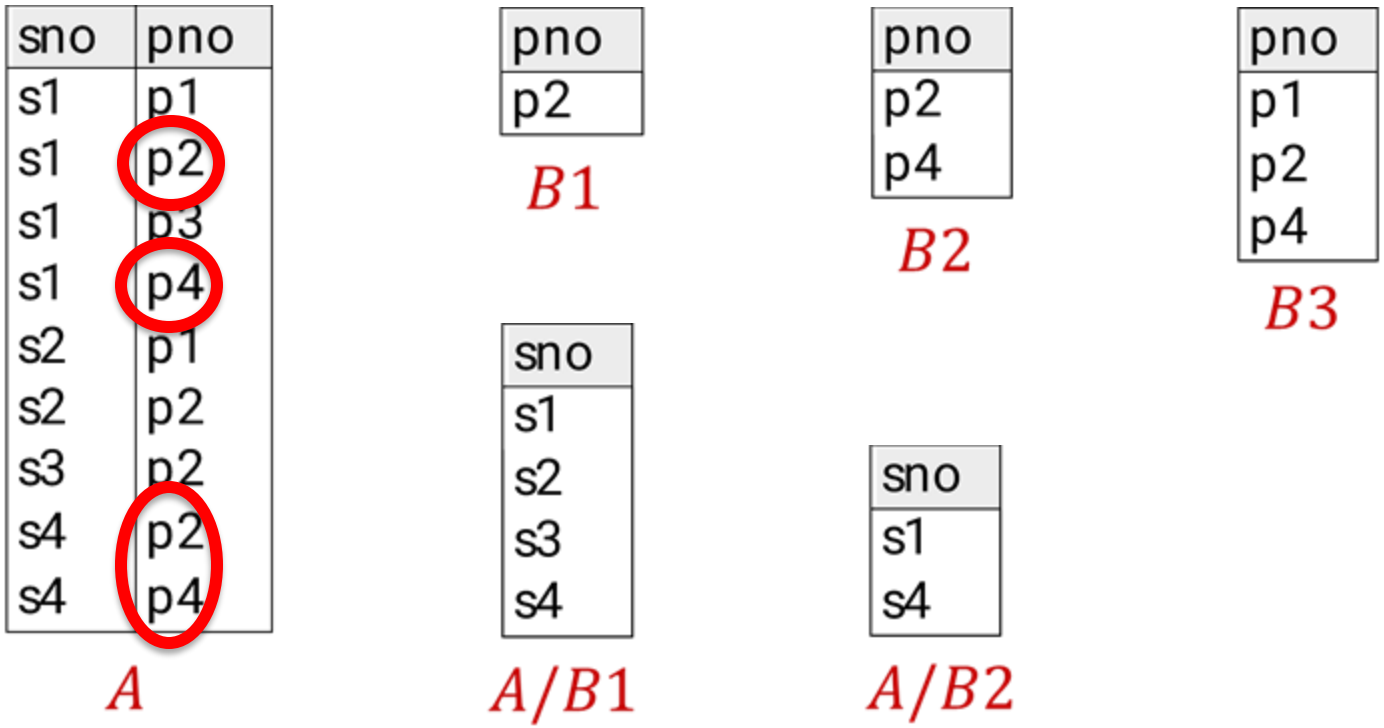
*there is no boat B without ...*

*a Reserves tuple showing S reserved B*

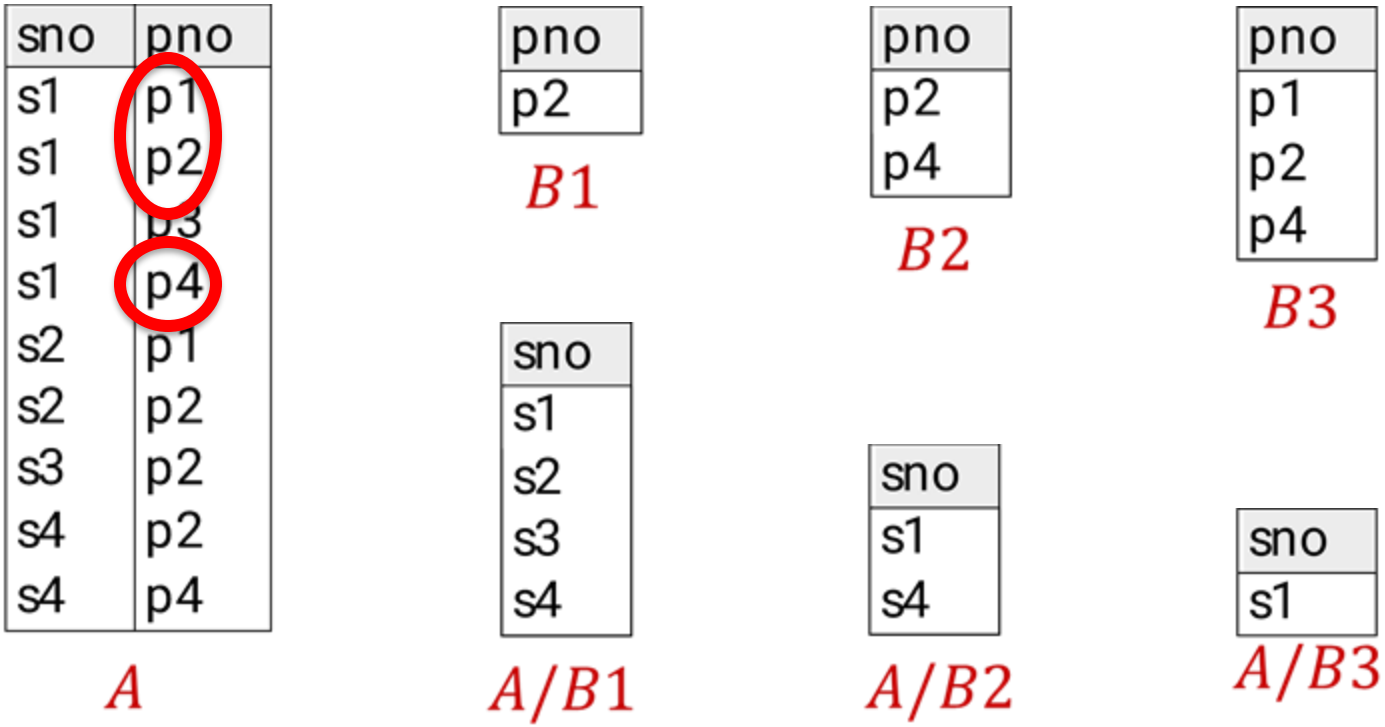
# Examples of Division A/B



# Examples of Division A/B



# Examples of Division A/B



# Expressing A/B:

$$\pi_{sno}(A) - \pi_{sno}((\pi_{sno}(A) \times B) - A)$$

| sno | pno |
|---|---|
| s1 | p1 |
| s1 | p2 |
| s1 | p3 |
| s1 | p4 |
| s2 | p1 |
| s2 | p2 |
| s3 | p2 |
| s4 | p2 |
| s4 | p4 |

$A$

| sno | pno |
|---|---|
| s1 | p1 |
| s1 | p2 |
| s1 | p4 |
| s2 | p1 |
| s2 | p2 |
| s2 | p4 |
| s3 | p1 |
| s3 | p2 |
| s3 | p4 |
| s4 | p1 |
| s4 | p2 |
| s4 | p4 |

$T1 = \pi_{sno}(A) \times B$

$\Leftarrow$

| sno |
|---|
| s1 |
| s2 |
| s3 |
| s4 |

$\pi_{sno}(A)$

$\times$

| pno |
|---|
| p1 |
| p2 |
| p4 |

$B$

# Expressing A/B: $\pi_{sno}(A) - \pi_{sno}(T1 - A)$

| sno | pno |
|---|---|
| s1 | p1 |
| s1 | p2 |
| s1 | p3 |
| s1 | p4 |
| s2 | p1 |
| s2 | p2 |
| s3 | p2 |
| s4 | p2 |
| s4 | p4 |

$A$

| sno | pno |
|---|---|
| ~~s1~~ | ~~p1~~ |
| ~~s1~~ | ~~p2~~ |
| ~~s1~~ | ~~p4~~ |
| ~~s2~~ | ~~p1~~ |
| ~~s2~~ | ~~p2~~ |
| s2 | p4 |
| s3 | p1 |
| ~~s3~~ | ~~p2~~ |
| s3 | p4 |
| s4 | p1 |
| ~~s4~~ | ~~p2~~ |
| ~~s4~~ | ~~p4~~ |

$T1 = \pi_{sno}(A) \times B$

| sno | pno |
|---|---|
| s2 | p4 |
| ~~s3~~ | ~~p1~~ |
| s3 | p4 |
| s4 | p1 |

$T1 - A$

| pno |
|---|
| p1 |
| p2 |
| p4 |

$B$

| sno |
|---|
| s2 |
| s3 |
| s4 |

$T2 = \pi_{sno}(T1 - A)$

# Expressing A/B: $\pi_{sno}(A) - T2$

| sno | pno |
|---|---|
| s1 | p1 |
| s1 | p2 |
| s1 | p3 |
| s1 | p4 |
| s2 | p1 |
| s2 | p2 |
| s3 | p2 |
| s4 | p2 |
| s4 | p4 |

$A$

| sno | pno |
|---|---|
| ~~s1~~ | ~~p1~~ |
| ~~s1~~ | ~~p2~~ |
| ~~s1~~ | ~~p4~~ |
| ~~s2~~ | ~~p1~~ |
| ~~s2~~ | ~~p2~~ |
| s2 | p4 |
| s3 | p1 |
| ~~s3~~ | ~~p2~~ |
| s3 | p4 |
| s4 | p1 |
| ~~s4~~ | ~~p2~~ |
| ~~s4~~ | ~~p4~~ |

$T1 = \pi_{sno}(A) \times B$

| sno | pno |
|---|---|
| s2 | p4 |
| ~~s3~~ | ~~p1~~ |
| s3 | p4 |
| s4 | p1 |

$T1 - A$

$\pi_{sno}($

| sno |
|---|
| s1 |
| s2 |
| s3 |
| s4 |

$-$

| sno |
|---|
| s2 |
| s3 |
| s4 |

$=$

| sno |
|---|
| s1 |

$T2 = \pi_{sno}(T1 - A)$

| pno |
|---|
| p1 |
| p2 |
| p4 |

$B$

$A/B = \pi_{sno}(A) - T2$

# The end!