

```
drop table reserves;
drop table sailors;
drop table boats;
```

--- create tables according to the definition chapter 5.1

```
create table sailors(
    sid    integer not null constraint sailors_pk primary key,
    sname  varchar(20),
    rating integer,
    age    integer
);
create table boats(
    bid    integer not null constraint boat_pk primary key,
    bname  varchar(20),
    color  varchar(20)
);
create table reserves(
    sid    integer,
    bid    integer,
    day    date,
    constraint reserves_pk primary key (sid,bid,day),
    constraint reserve_sailor_fk foreign key (sid) references sailors(sid),
    constraint reserve_boat_fk foreign key (bid) references boats(bid)
);
```

--- popula as tabelas

```
insert into sailors (sid,sname,rating,age) values(22,'Dustin',7,45.0);
insert into sailors (sid,sname,rating,age) values(29,'Brutus',1,33.0);
insert into sailors (sid,sname,rating,age) values(31,'Lubber',8,55.5);
insert into sailors (sid,sname,rating,age) values(32,'Andy',8,25.5);
insert into sailors (sid,sname,rating,age) values(58,'Rusty',10,35.0);
insert into sailors (sid,sname,rating,age) values(64,'Horataio',7,35.0);
insert into sailors (sid,sname,rating,age) values(71,'Zorba',10,16.0);
insert into sailors (sid,sname,rating,age) values(74,'Horataio',9,35.0);
insert into sailors (sid,sname,rating,age) values(85,'Art',3,25.5);
insert into sailors (sid,sname,rating,age) values(95,'Bob',3,63.5);
```

```
insert into boats (bid,bname,color) values(101,'Interlake','blue');
insert into boats (bid,bname,color) values(102,'Interlake','red');
insert into boats (bid,bname,color) values(103,'Clipper','green');
insert into boats (bid,bname,color) values(104,'Marine','red');
```

```
insert into reserves(sid,bid,day) values(22,101,'10-10-1998');
insert into reserves(sid,bid,day) values(22,102,'10-10-1998');
insert into reserves(sid,bid,day) values(22,103,'8-10-1998');
insert into reserves(sid,bid,day) values(22,104,'7-10-1998');
insert into reserves(sid,bid,day) values(31,102,'10-11-1998');
insert into reserves(sid,bid,day) values(31,103,'6-11-1998');
insert into reserves(sid,bid,day) values(31,104,'12-11-1998');
insert into reserves(sid,bid,day) values(64,101,'5-11-1998');
insert into reserves(sid,bid,day) values(64,102,'8-09-1998');
```

-- Q1 "Find the names of sailors who have reserved boat number 103"

[illegible]

REM Q3 "Find the colors of boats reserved by Lubber"

```
SELECT B.color
FROM Sailors S NATURAL JOIN Reserves R NATURAL JOIN Boats B
WHERE S.sname ='Lubber';
```

REM Q4 "Find the names of sailors who have reserved at least one boat"

```
SELECT S.sname
FROM Sailors S NATURAL JOIN Reserves;
```

```
SELECT DISTINCT S.sname
FROM Sailors S NATURAL JOIN Reserves R;
```

REM Q5 "Find the names of sailors who have reserved a red or a gree boat"

```
SELECT S.sname
FROM Sailors S NATURAL JOIN Reserves R NATURAL JOIN Boats B
WHERE B.color='red' OR B.color='green';
```

```
SELECT S.sname
FROM Sailors S NATURAL JOIN Reserves R NATURAL JOIN Boats B
WHERE B.color in ('red','green');
```

```
SELECT S.sname
FROM Sailors S NATURAL JOIN Reserves R NATURAL JOIN Boats B
WHERE B.color='red'
UNION
SELECT S2.sname
FROM Sailors S2 NATURAL JOIN Boats B2 NATURAL JOIN Reserves R2
WHERE B2.color = 'green';
```

REM Q6 "Find the names of sailors who have reserved both a red and a green boat"

```
SELECT S.sname
FROM Sailors S NATURAL JOIN Reserves R NATURAL JOIN Boats B
WHERE B.color = 'red'
INTERSECT
SELECT S2.sname
FROM Sailors S2 NATURAL JOIN Boats B2 NATURAL JOIN Reserves R2
WHERE B2.color= 'green';
```

REM Nested Query

```
SELECT S.sname
FROM Sailors S NATURAL JOIN Reserves R NATURAL JOIN Boats B
WHERE B.color ='red'
AND S.sid IN (SELECT S2.sid
              FROM Sailors S2 NATURAL JOIN Boats B2 NATURAL JOIN Reserves R2
              WHERE B2.color ='green');
```

REM **"Find the names of sailors who have reserved a red but not a green boat"**
REM

```
SELECT S.sname
FROM Sailors S NATURAL JOIN Reserves R NATURAL JOIN Boats B
WHERE B.color = 'red'
EXCEPT
SELECT S2.sname
FROM Sailors S2 NATURAL JOIN Boats B2 NATURAL JOIN Reserves R2
WHERE B2.color= 'green';
```

```
SELECT S.sname
FROM Sailors S NATURAL JOIN Reserves R NATURAL JOIN Boats B
WHERE B.color ='red'
AND S.sid NOT IN (SELECT S2.sid
                  FROM Sailors S2 NATURAL JOIN Boats B2 NATURAL JOIN Reserves R2
                  WHERE B2.color ='green');
```

REM Q7 **"Find the names of sailors who have reserved at least two different boats"**

```
SELECT DISTINCT S.sname
FROM Sailors S NATURAL JOIN Reserves R1 JOIN Reserves R2 ON R1.sid = R2.sid AND
R1.bid != R2.bid;
```

REM **"Find the names of sailors who have reserved at least n boats"**
REM THE SAME IDEA IS TO JOIN N RELATIONS --- TOO DEDIOUS
REM We can do this by combining CNT, GROUP BY, and nested query together.
REM The question is how we can do this before we adress GROUP BY.
REM Assume one dbms does not support GROUP BY and HAVING, how will you help
REM them implement this? HINT: the same relation equ-join many times.

REM IS THERE ANY DIFFERENCE BETWEEN THE TWO FOLLOWING EXPRESSION?
REM IS the next one the same as the above one?
REM INSERT INTO Reserves Values(74,103,'08-DEC-98');

```
SELECT S.sname
from Sailors S NATURAL JOIN Reserves R
GROUP BY S.sname
HAVING COUNT(*) > 1;
```

```
SELECT S1.sname
FROM Sailors S1
WHERE S1.sid IN (
    SELECT S.sid
    from Sailors S NATURAL JOIN Reserves R
    GROUP BY S.sid
    HAVING COUNT(*) > 1);
```


REM Q11 "Find all sailors with a rating above 7"

```
SELECT S.sid, S.sname, S.rating, S.age
FROM Sailors S
WHERE S.rating > 7;
```

REM Q12 "Find the names and ages of sailors with a rating above 7"

```
SELECT S.Sname, S.age
FROM Sailors S
WHERE S.rating > 7;
```

REM Q15 "Find the names and ages of all sailors"

```
SELECT DISTINCT S.sname, S.age
FROM Sailors S;
```

REM Q16 "Find the sids of sailors who have reserved a red boat";

```
SELECT R.sid
FROM Boats B NATURAL JOIN Reserves R
WHERE B.color = 'red';
```

REM Q18 Find the ages of sailors whose name begins and ends with B and has at least three characters

```
SELECT S.age
FROM Sailors S
WHERE S.sname LIKE 'B_%B';
```

REM Q19 Find the sids of all sailors who have reserved red boats but not green boats

```
SELECT S.sid
FROM Sailors S NATURAL JOIN Reserves R NATURAL JOIN Boats B
WHERE B.color = 'red'
EXCEPT
SELECT S2.sid
FROM Sailors S2 NATURAL JOIN Reserves R2 NATURAL JOIN Boats B2
WHERE B2.color = 'green';
```

```
SELECT R.sid
FROM Boats B NATURAL JOIN Reserves R
WHERE B.color = 'red'
EXCEPT
SELECT R2.sid
FROM Boats B2 NATURAL JOIN Reserves R2
WHERE B2.color = 'green';
```

REM Q21 "Find the names of sailors who have not reserved a red boat"

```
SELECT S.sname
FROM Sailors S
WHERE S.sid NOT IN (SELECT R.sid
                    FROM Reserves R
                    WHERE R.bid IN (SELECT B.bid
                                    FROM Boats B
                                    WHERE B.color='red'));
```

REM Q22 "Find sailors whose rating is better than some sailor called Horatio"

REM SET comparsion operators

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

REM Q23 "Find sailors whose rating is better than every sailor called Horatio"

```
SELECT S.sid
FROM Sailors S
WHERE S.rating > ALL(SELECT S2.rating
                    FROM Sailors S2
                    WHERE S2.sname = 'Horatio');
```

REM Q24 "Find the sailors with the highest rating"

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

REM Q25 "Find the average of all sailors"

```
SELECT AVG (S.age)
FROM Sailors S;
```

REM Q26 "Find the average age of sailors with a rating of 10"

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

REM Q27 "Find the name and age of the oldest sailor"

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

REM Q28 "Count the number of sailors"

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

REM Q29 "Count the number of different sailor names"

```
SELECT COUNT (DISTINCT S.sname)  
FROM Sailors S;
```

REM Q30 "Find the names of sailors who are older than the oldest sailor with a rating of 10"

```
SELECT S.sname  
FROM Sailors S  
WHERE S.age > (SELECT MAX(S2.age)  
               FROM Sailors S2  
               WHERE S2.rating = 10);
```

```
SELECT S.sname  
FROM Sailors S  
WHERE S.age > ALL (SELECT S2.age  
                  FROM Sailors S2  
                  WHERE S2.rating = 10);
```

REM Q31 "Find the age of the youngest sailor for each rating level"

```
SELECT S.rating, MIN(S.age)  
FROM Sailors S  
GROUP BY S.rating ;
```

REM Q32 "Find the age of the youngest sailor who is eligible to vote (i.e., is at least 18 years old) for each rating level with at least two such sailors"

```
SELECT S.rating, MIN(S.age) AS minage  
FROM Sailors S  
WHERE S.age >=18  
GROUP BY S.rating  
HAVING COUNT(*) > 1;
```

REM Q33 "For each red boat, find the number of reservations for this boat"

```
SELECT B.bid, COUNT(*) AS sailorcount  
FROM Boats B NATURAL JOIN Reserves R  
WHERE B.color = 'red'  
GROUP BY B.bid;
```


REM Q34 "Find the average age of sailors for each rating level that has at least two sailors"

```
SELECT S.rating, AVG(S.age) AS average
FROM Sailors S
GROUP BY S.rating
HAVING COUNT(*) > 1;
```

REM Q37 "Find those ratings for which the average age of sailors is the minimum over all ratings"

```
SELECT Temp.rating, Temp.average
FROM (SELECT S.rating, AVG(S.age) AS average
      FROM Sailors S
      GROUP BY S.rating) Temp
WHERE Temp.average = (SELECT MIN(Temp.average) FROM Temp);
```