Solution to queries and updates in SQL assignment

The boat reservations database has the following schema:

sailor: sname (string), rating (integer)
boat: bname (string), color (string), rating (integer)
reservation: sname (string), bname (string), day (string)

The rating attribute for boats indicate the minimum rating required of a sailor reserving the boat.
Use SQL to do the following:

(d) Formulate and execute the following queries using SQL:

1. List all boats reserved on Wednesday and their color.

   select reservation.bname, color
   from reservation, boat
   where reservation.bname = boat.bname and reservation.day = 'Wednesday'

2. List the sailors with the highest rating. Provide two queries:
   (i) one using the MAX aggregate function, and

      select sname
      from sailor
      where rating in
      (select MAX(rating) from sailor)

   (ii) another without using MAX.

      select s.sname
      from sailor s
      where not exists
      (select *
       from sailor
       where rating > s.rating)

   Or:
select sname
from sailor
where rating not in
  (select x.rating from sailor x, sailor y
   where x.rating < y.rating )

3. List all pairs of sailors who have reserved boats on the same day (avoid listings of the form \((a, a)\), or listing both \((a, b)\) and \((b, a)\)).

select x.sname, y.sname
from reservation x, reservation y
where x.day = y.day and x.sname < y.sname

4. For each day, list the number of red boats reserved on that day.

   (select r.day, count(r.boat) as number
from reservation r, boat b
where r.bname = b.bname and b.color = 'red'
group by day )
union
   (select day, 0 as number
from reservation
where day not in
   (select day
    from reservation, boat
    where reservation.bname = boat.bname and color = 'red') )

The second query in the union forces the display of a count of zero for days when no red boats are reserved.

5. List the days when only red boats are reserved.

select day
from reservation
where day not in
   (select r.day
    from reservation r, boat b
    where r.bname = b.bname and b.color \neq \text{'red'} )

This makes the assumption that each boat has only one color.
The answer on the sample data is Sunday, Monday. Without the assumption we would write:

```sql
select day 
from reservation 
where day not in 
  (select r.day from reservation r where r.bname not in 
  (select bname from boat where color = 'red'))
```

6. List the days when no red boat is reserved.

```sql
select r.day 
from reservation r 
where not exists 
  (select * from reservation, boat 
   where day = r.day and reservation.bname = boat.bname and color = 'red')
```

The answer for the sample data is Saturday.

7. List the days when all red boats are reserved. Provide three SQL queries, using nested sub-queries in different ways:

- with NOT IN tests;

```sql
select day 
from reservation 
where day not in 
  (select r.day from reservation r, boat b 
   where b.color = 'red' and r.day not in 
   (select day from reservation 
    where bname = b.bname ))
```

The answer for the sample data is Monday, Wednesday.

- with NOT EXISTS tests;
select x.day  
from reservation x  
where not exists  
  (select * from boat b  
   where b.color = 'red' and not exists  
     (select * from reservation r  
      where r.day = x.day and r.bname = b.bname ))  
- with COUNT aggregate functions.

select x.day  
from reservation x  
where  
  (select count(*) from boat b  
   where b.color = 'red' and  
     (select count(*) from reservation r  
      where r.day = x.day and r.bname = b.bname) = 0  
  ) = 0

8. Evaluate the query above by creating temporary views to break down the query into simpler queries and avoid nested sub-queries as much as possible (explain the meaning of each view you create).

Here is one possibility:
create view redboat as  
select bname from boat  
where color = 'red'

create view notallreds as  
select day from reservation, redboat  
where day not in  
  (select day from reservation  
   where bname = redboat.bname)

select day from reservation  
where day not in  
  (select * from notallreds)

9. For each day of the week, list the average rating of sailors having reserved boats that day.
create view unique as
select distinct sailor.sname, rating, day
from sailor, reservation
where sailor.sname = reservation.sname;
select day, avg(rating) = avg(rating * 1.0)
from unique
group by day

The reason for multiplying rating with 1.0 in \text{avg}(\text{rating} * 1.0) is to convert the rating to a real so that the average is not truncated to an integer. The role of the view \textit{unique} is to provide the list of sailors and their ratings for each day \textit{without duplicates}. This is to avoid counting the same sailor multiple times if the sailor has multiple reservations in the same day. For example, the following alternative does not avoid this problem:

\begin{verbatim}
select day, avg(rating) = avg(rating)
from sailor, reservation
where sailor.sname = reservation.sname
group by day
\end{verbatim}

(e) Formulate and run a query to verify that all sailors having reservations are qualified to sail the boats they reserved. Specifically, the query should look for violations. It should return the sailor/boat pairs that violate the constraint together with the rating of the sailor, the rating of the boat, and the day of the reservation. The answer should be empty if there is no violation.

\begin{verbatim}
select r.day, s.sname, b.bname, s.rating, b.rating
from sailor s, reservation r, boat b
where s.sname = r.sname and r.bname = b.bname and s.rating < b.rating
\end{verbatim}

(f) Formulate and execute the following updates in SQL.

1. Switch all Wednesday and Monday reservations, without explicitly naming the boats involved.
update reservation
set day = 'xxx' where day = 'Monday';

update reservation
set day = 'Monday'
where day = 'Wednesday';

update reservation
set day = 'Wednesday'
where day = 'xxx'

2. Delete all reservations violating the constraint in (e) above.

delete from reservation
where exists (select * from sailor s, boat b
where s.sname=reservation.sname and b.bname = reservation.bname
and s.rating < b.rating)