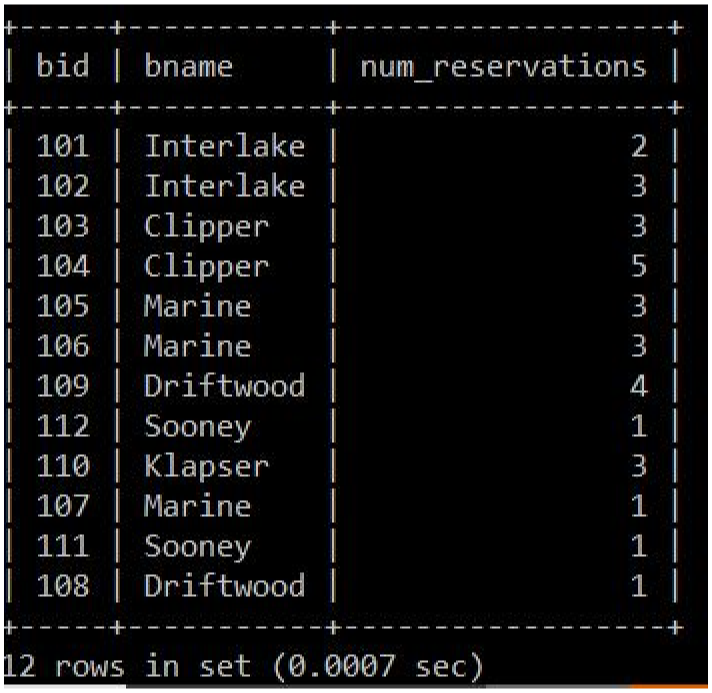
Omar Thenmalai **10/19/2020**
Databases **Problem Set #1**

Part 1

1. List, for every boat, the number of times it has been reserved, excluding those boats that have never been reserved (list the id and the name).

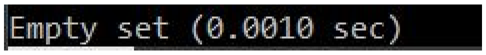
```
SELECT boats.bid, boats.bname, COUNT(*) as num_reservations
FROM boats, reserves
WHERE boats.bid = reserves.bid
GROUP BY boats.bid
HAVING num_reservations > 0;
```



```
+-----+-----------+------------------+
| bid | bname     | num_reservations |
+-----+-----------+------------------+
| 101 | Interlake |                2 |
| 102 | Interlake |                3 |
| 103 | Clipper   |                3 |
| 104 | Clipper   |                5 |
| 105 | Marine    |                3 |
| 106 | Marine    |                3 |
| 109 | Driftwood |                4 |
| 112 | Sooney    |                1 |
| 110 | Klapser   |                3 |
| 107 | Marine    |                1 |
| 111 | Sooney    |                1 |
| 108 | Driftwood |                1 |
+-----+-----------+------------------+
12 rows in set (0.0007 sec)
```

2. List those sailors who have reserved every red boat (list the id and the name).

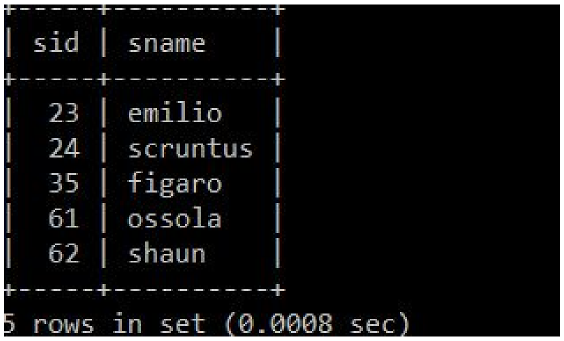
```
SELECT sailors.sid, sailors.sname
FROM sailors , (
     SELECT reserves.sid,COUNT(DISTINCT(reserves.bid)) AS
count_of_red
     FROM reserves, boats, sailors
     WHERE reserves.bid = boats.bid
     AND reserves.sid = sailors.sid
     AND boats.color = 'red'
     GROUP BY reserves.sid
) AS temp
WHERE sailors.sid = temp.sid
AND temp.count_of_red = (SELECT COUNT(*)
                         FROM boats
                         WHERE boats.color = 'red'
);
```



```
Empty set (0.0010 sec)
```

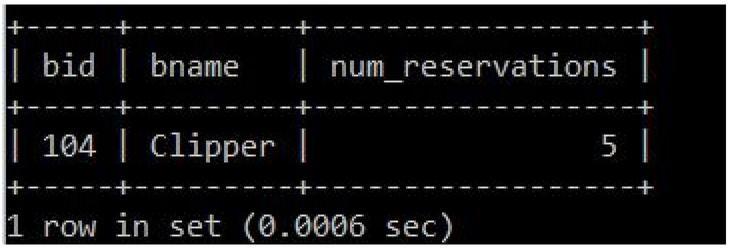
3. List those sailors who have reserved only red boats.

```
SELECT DISTINCT sailors.sid, sailors.sname
FROM sailors, boats, reserves
WHERE sailors.sid = reserves.sid
AND reserves.bid = boats.bid
AND boats.color = 'red'
AND sailors.sid NOT IN ( SELECT sailors.sid
                         FROM sailors, boats, reserves
                         WHERE sailors.sid = reserves.sid
                         AND boats.bid = reserves.bid
                         AND boats.color != 'red'
);
```


```
+-----+----------+
| sid | sname    |
+-----+----------+
|  23 | emilio   |
|  24 | scruntus |
|  35 | figaro   |
|  61 | ossola   |
|  62 | shaun    |
+-----+----------+
5 rows in set (0.0008 sec)
```

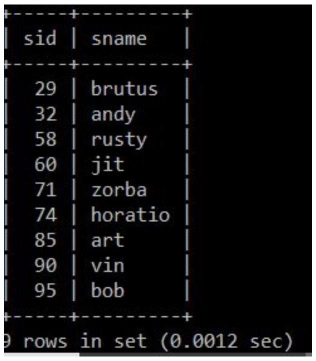
4. For which boat are there the most reservations?

```
SELECT boats.bid, boats.bname, COUNT(*) as num_reservations
FROM boats, reserves
WHERE boats.bid = reserves.bid
GROUP BY boats.bid
ORDER BY num_reservations DESC
LIMIT 1;
```


```
+-----+---------+------------------+
| bid | bname   | num_reservations |
+-----+---------+------------------+
| 104 | Clipper |                5 |
+-----+---------+------------------+
1 row in set (0.0006 sec)
```

5. Select all sailors who have never reserved a red boat.

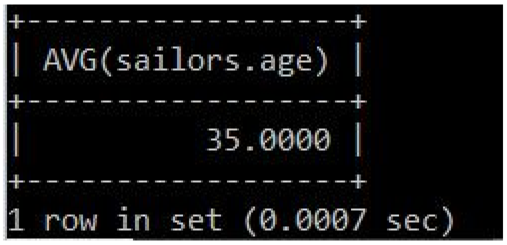
```
SELECT DISTINCT sailors.sid as sid, sailors.sname
FROM sailors, boats, reserves
WHERE sailors.sid = reserves.sid
AND reserves.bid = boats.bid
AND boats.color != ‘red’
AND sailors.sid NOT IN ( SELECT sailors.sid
                    FROM sailors, boats, reserves
                    WHERE sailors.sid = reserves.sid
                    AND boats.bid = reserves.bid
                    AND boats.color = ‘red’ )
UNION
SELECT DISTINCT sailors.sid, sailors.sname
FROM sailors, reserves
WHERE sailors.sid NOT IN ( SELECT reserves.sid from reserves)
ORDER BY sid;
```



```
+-----+---------+
| sid | sname   |
+-----+---------+
|  29 | brutus  |
|  32 | andy    |
|  58 | rusty   |
|  60 | jit     |
|  71 | zorba   |
|  74 | horatio |
|  85 | art     |
|  90 | vin     |
|  95 | bob     |
+-----+---------+
9 rows in set (0.0012 sec)
```

6. Find the average age of sailors with a rating of 10.

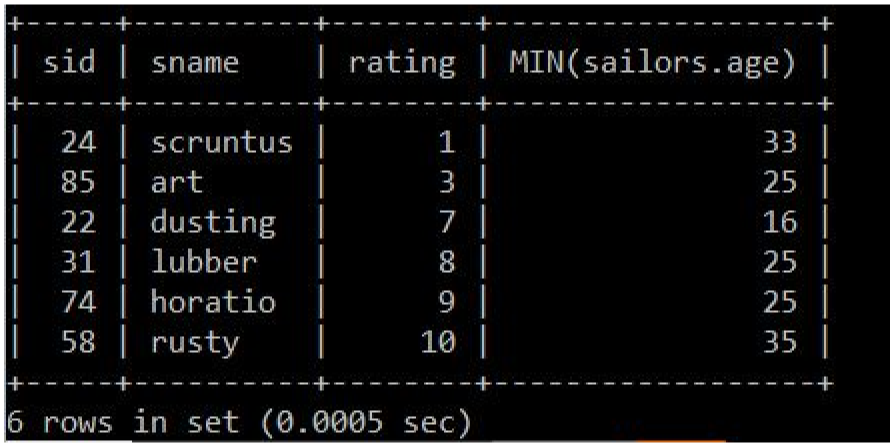
```
SELECT AVG(sailors.age)
FROM sailors
WHERE sailors.rating = 10;
```



```
+------------------+
| AVG(sailors.age) |
+------------------+
|          35.0000 |
+------------------+
1 row in set (0.0007 sec)
```

7. For each rating, find the name and id of the youngest sailor.

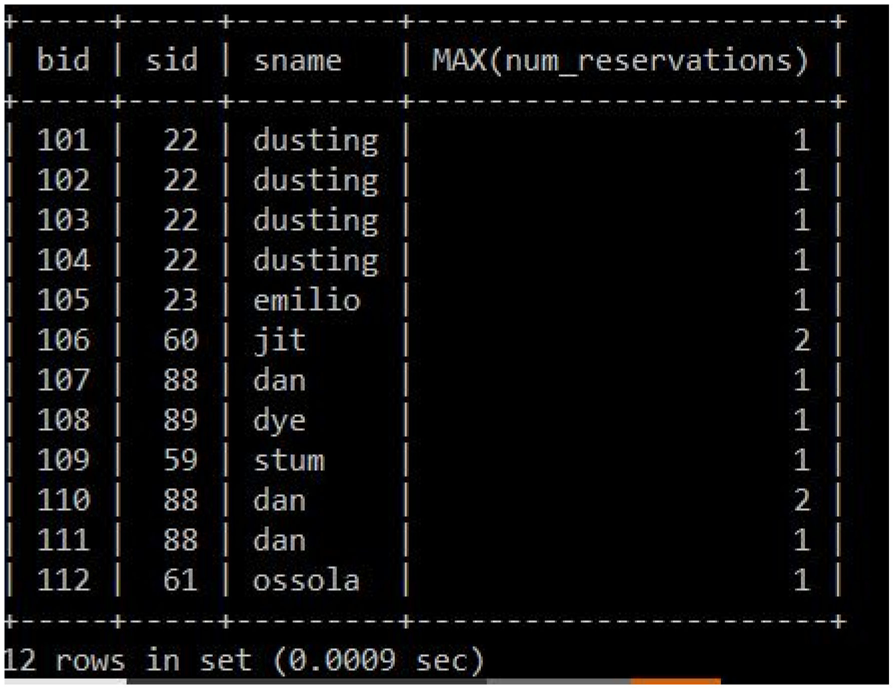
```
SELECT sailors.sid, sailors.sname, sailors.rating,
MIN(sailors.age)
FROM sailors
GROUP BY sailors.rating
ORDER BY sailors.rating;
```



```
+-----+----------+--------+------------------+
| sid | sname    | rating | MIN(sailors.age) |
+-----+----------+--------+------------------+
|  24 | scruntus |      1 |               33 |
|  85 | art      |      3 |               25 |
|  22 | dusting  |      7 |               16 |
|  31 | lubber   |      8 |               25 |
|  74 | horatio  |      9 |               25 |
|  58 | rusty    |     10 |               35 |
+-----+----------+--------+------------------+
6 rows in set (0.0005 sec)
```

8. Select, for each boat, the sailor who made the highest number of reservations for that boat.

```
SELECT c.bid, c.sid, c.sname, MAX(num_reservations)
FROM ( SELECT sailors.sname, reserves.sid, reserves.bid,
          COUNT(reserves.bid) as num_reservations
          FROM sailors, reserves
          WHERE sailors.sid = reserves.sid
          GROUP BY reserves.bid, reserves.sid
          ORDER BY num_reservations DESC
     ) as c
GROUP BY c.bid
ORDER BY c.bid;
```



```
+-----+-----+---------+-----------------------+
| bid | sid | sname   | MAX(num_reservations) |
+-----+-----+---------+-----------------------+
| 101 |  22 | dusting |                     1 |
| 102 |  22 | dusting |                     1 |
| 103 |  22 | dusting |                     1 |
| 104 |  22 | dusting |                     1 |
| 105 |  23 | emilio  |                     1 |
| 106 |  60 | jit     |                     2 |
| 107 |  88 | dan     |                     1 |
| 108 |  89 | dye     |                     1 |
| 109 |  59 | stum    |                     1 |
| 110 |  88 | dan     |                     2 |
| 111 |  88 | dan     |                     1 |
| 112 |  61 | ossola  |                     1 |
+-----+-----+---------+-----------------------+
12 rows in set (0.0009 sec)
```

```
/* If there is a tie for the max number of reservations, take
the sailor with the lowest sid number */
```

Part 2

I wrote a test for each of the questions in part 1. Each test uses the SQL query that was written for a question in part 1 and compares the results with that of the ORM query. I used *SQLAlchemy* as my ORM and implemented the tests using *pytest*.

Part 3

Using the SQL script "part3_sailors-mysql.sql", two new tables were added to the dataset: *employees*, and *shifts*. The schema for these two tables is as follows:

```
employees
| Field       | Type     | Null | Key | Default | Extra |
+-------------+----------+------+-----+---------+-------+
| eid         | int      | NO   | PRI | NULL    |       |
| ename       | char(20) | YES  |     | NULL    |       |
| hourly_wage | int      | YES  |     | NULL    |       |
+-------------+----------+------+-----+---------+-------+
```

```
shifts
+-------------+----------+------+-----+---------+-------+
| Field       | Type     | Null | Key | Default | Extra |
+-------------+----------+------+-----+---------+-------+
| eid         | int      | NO   | PRI | NULL    |       |
| shift_start | datetime | NO   | PRI | NULL    |       |
| shift_end   | datetime | NO   | PRI | NULL    |       |
+-------------+----------+------+-----+---------+-------+
```

Every employee has an *hourly_wage*, which can be used in combination with the *shift_start* and *shift_end* to determine how much they should be paid for a given day.

The *boats* table was also updated using an updated version of "sailors-mysql.sql" to include the daily cost to the owners from maintenance, and the daily price to the customers. As was done with the original dataset, I assume that the boats are rented on a by-day basis.

boats

```
+-------------+----------+------+-----+---------+-------+
| Field       | Type     | Null | Key | Default | Extra |
+-------------+----------+------+-----+---------+-------+
| bid         | int      | NO   | PRI | NULL    |       |
| bname       | char(20) | YES  |     | NULL    |       |
| color       | char(10) | YES  |     | NULL    |       |
| length      | int      | YES  |     | NULL    |       |
| daily_price | int      | YES  |     | NULL    |       |
| daily_cost  | int      | YES  |     | NULL    |       |
+-------------+----------+------+-----+---------+-------+
```

Taken together with the *reserves* table, the new information can be used by the owners to calculate their profit for a given day.

Functionality was also implemented for the owners to change the hourly wage of a given employee. The owner can also change the *daily_price* and *daily_cost* for a given boat.

Tests were written that verify the various functions, and ensure that the respective tables are properly updated if the owner were to make a change to an employee's *hourly_wage*, or a boat's *daily_price* or *daily_cost*.