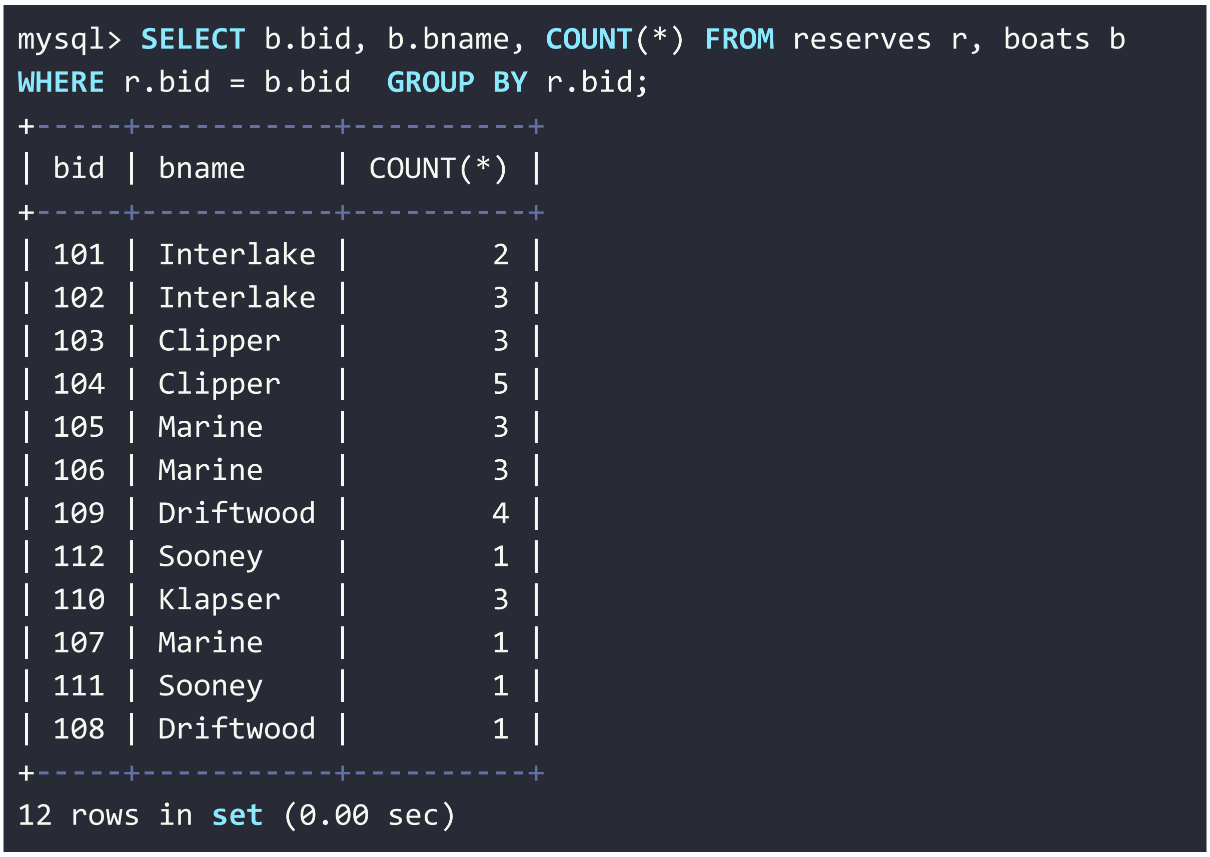
Thodori Kapouranis

Professor Eugene Sokolov

Databases Problem Set 1

Part 1

Q1 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).



```
mysql> SELECT b.bid, b.bname, COUNT(*) FROM reserves r, boats b
WHERE r.bid = b.bid  GROUP BY r.bid;
+-----+-----------+----------+
| bid | bname     | COUNT(*) |
+-----+-----------+----------+
| 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 (0.00 sec)
```

Q2 List the names of the sailors that have rented every red boat.

```
mysql> SELECT s.sname, s.sid FROM sailors s, reserves r, (SELECT *
FROM boats b WHERE b.color='red') rb WHERE s.sid=r.sid AND r.bid =
rb.bid GROUP BY r.bid HAVING COUNT(r.bid)=(SELECT count(*) FROM
boats b WHERE b.color='red');

Empty set (0.00 sec)
```

Q3. List those sailors who have reserved only red boats.

```
mysql> SELECT distinct s.sid, s.sname FROM sailors s, reserves r,
boats b WHERE r.sid NOT IN (SELECT r2.sid FROM reserves r2, boats
b2 WHERE r2.bid=b2.bid AND (b2.color='green' OR b2.color='blue'))
AND r.sid=s.sid;
+-----+----------+
| sid | sname    |
+-----+----------+
|  23 | emilio   |
|  24 | scruntus |
|  35 | figaro   |
|  61 | ossola   |
|  62 | shaun    |
+-----+----------+
5 rows in set (0.00 sec)
```

Q4. For which boat are there the most reservations?

```
mysql> SELECT r.bid, b.bname, COUNT(r.bid) total FROM reserves r,
boats b WHERE r.bid=b.bid GROUP BY r.bid HAVING total=(SELECT
COUNT(r.bid) total FROM reserves r, boats b WHERE r.bid=b.bid
GROUP by r.bid ORDER BY total DESC LIMIT 1);

+-----+---------+-------+
| bid | bname   | total |
+-----+---------+-------+
| 104 | Clipper |     5 |
+-----+---------+-------+
1 row in set (0.00 sec)
```

Q5. Select all sailors who have never reserved a red boat.

```
mysql> SELECT distinct s.sid, s.sname FROM sailors s, reserves r,
boats b WHERE s.sid NOT IN (SELECT distinct s2.sid FROM sailors
s2, reserves r2, boats b2 WHERE s2.sid=r2.sid AND r2.bid=b2.bid
AND b2.color='red') ORDER BY (s.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 (0.00 sec)
```

Q6. Find the average age of sailors with a rating of 10.

```
mysql> SELECT AVG(s.age) from sailors s WHERE s.rating=10
    -> ;
+------------+
| AVG(s.age) |
+------------+
|    35.0000 |
+------------+
1 row in set (0.02 sec)
```

Q7. For each rating, find the name and id of the youngest sailor.

```
mysql> SELECT s.rating, s.sid, s.sname, s.age FROM sailors s WHERE
s.age=( SELECT MIN(s2.age) FROM sailors s2 WHERE
s2.rating=s.rating) ORDER BY s.rating;
+--------+-----+----------+------+
| rating | sid | sname    | age  |
+--------+-----+----------+------+
|      1 |  24 | scruntus |   33 |
|      1 |  29 | brutus   |   33 |
|      3 |  85 | art      |   25 |
|      3 |  89 | dye      |   25 |
|      7 |  61 | ossola   |   16 |
|      7 |  64 | horatio  |   16 |
|      8 |  32 | andy     |   25 |
|      8 |  59 | stum     |   25 |
|      9 |  74 | horatio  |   25 |
|      9 |  88 | dan      |   25 |
|     10 |  58 | rusty    |   35 |
|     10 |  60 | jit      |   35 |
|     10 |  62 | shaun    |   35 |
|     10 |  71 | zorba    |   35 |
+--------+-----+----------+------+
14 rows in set (0.00 sec)
```

Q8 Select, for each boat, the sailor who made the highest number of reservations for that boat.

```
mysql> CREATE TEMPORARY TABLE temp1
    -> SELECT r.sid, r.bid, b.bname, COUNT(*) as res
    -> FROM reserves r, boats b WHERE r.bid = b.bid
    -> GROUP BY b.bid, r.sid;

mysql> CREATE TEMPORARY TABLE  temp2
    -> SELECT temp1.bid, MAX(temp1.res) maxres FROM temp1 GROUP BY
temp1.bid;

mysql> SELECT t1.bname, t1.bid, s.sname, t1.sid, t1.res FROM
sailors s, temp1 t1, temp2 t2 WHERE t1.bid=t2.bid AND
t1.res=t2.maxres AND s.sid=t1.sid ORDER BY t1.bid;
+-----------+-----+----------+-----+-----+
| bname     | bid | sname    | sid | res |
+-----------+-----+----------+-----+-----+
| Interlake | 101 | dusting  |  22 |   1 |
| Interlake | 101 | horatio  |  64 |   1 |
| Interlake | 102 | dusting  |  22 |   1 |
| Interlake | 102 | lubber   |  31 |   1 |
| Interlake | 102 | horatio  |  64 |   1 |
| Clipper   | 103 | dusting  |  22 |   1 |
| Clipper   | 103 | lubber   |  31 |   1 |
| Clipper   | 103 | horatio  |  74 |   1 |
| Clipper   | 104 | dusting  |  22 |   1 |
| Clipper   | 104 | emilio   |  23 |   1 |
| Clipper   | 104 | scruntus |  24 |   1 |
| Clipper   | 104 | lubber   |  31 |   1 |
| Clipper   | 104 | figaro   |  35 |   1 |
| Marine    | 105 | emilio   |  23 |   1 |
| Marine    | 105 | figaro   |  35 |   1 |
| Marine    | 105 | stum     |  59 |   1 |
| Marine    | 106 | jit      |  60 |   2 |
| Marine    | 107 | dan      |  88 |   1 |
| Driftwood | 108 | dye      |  89 |   1 |
| Driftwood | 109 | stum     |  59 |   1 |
| Driftwood | 109 | jit      |  60 |   1 |
| Driftwood | 109 | dye      |  89 |   1 |
| Driftwood | 109 | vin      |  90 |   1 |
| Klapser    | 110 | dan        |  88 |    2 |
| Sooney     | 111 | dan        |  88 |    1 |
| Sooney     | 112 | ossola     |  61 |    1 |
+------------+-----+------------+-----+------+
26 rows in set (0.00 sec)
```

Part 2

Results here, code in actual python file.

```
C:\Users\Thodori.DESKTOP-ICL6JJH\Documents\##Projects\ECE464-Databases\h
omework-1>py -m pytest test-part-2.py
========================================================================
====== test session starts
========================================================================
======
platform win32 -- Python 3.8.1, pytest-6.2.5, py-1.10.0, pluggy-1.0.0
rootdir:
C:\Users\Thodori.DESKTOP-ICL6JJH\Documents\##Projects\ECE464-Databases\h
omework-1
collected 8 items

test-part-2.py ........
[100%]

========================================================================
======= 8 passed in 1.84s
========================================================================
=======

C:\Users\Thodori.DESKTOP-ICL6JJH\Documents\##Projects\ECE464-Databases\h
omework-1>
```

Part 3

For this part I added a repair history table to keep track of repairs done to boats. I used this SQL query to create the table:

```
CREATE TABLE repairs (
    rid int,
    bid int,
    day date,
    cost DECIMAL(10,2),
        PRIMARY KEY (rid)
);
```

I decided to use Decimal values for the cost since most currencies in the world to my knowledge only represent money up to the 2nd decimal point. This avoids users from accidentally submitting an invalid money amount. I assumed dollars for my test input.

Although a repair-id primary key is not necessarily required for this table by itself, I thought it would be useful in the case that the cost column expands into its own repair-costs table and needs a way to be indexed.

For generating random inputs I just manually hardcoded some entries to populate the table.

I tested 3 queries using this repairs table:

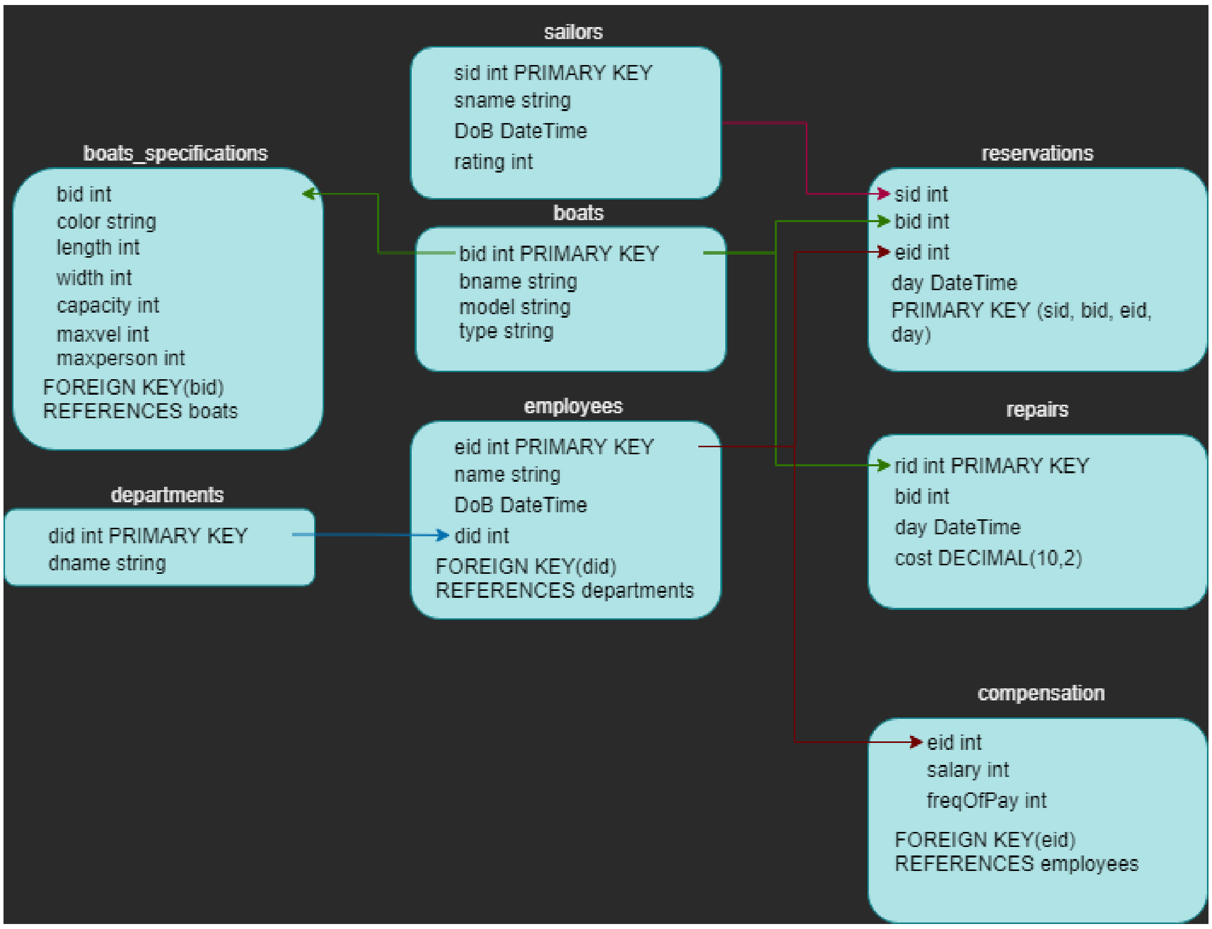
1. Select the boat with the highest repair cost
2. Select the boats that have not needed repairs
3. Get total repair costs for a certain date (October 2000)

I eyeballed the solutions given the small dataset and used the sqlalchemy ORM and pytest to assert that the ORM queries are working. All the tests passed.

Additional things can be changed to the original database to make it better. For one, the sailors' age should rather be the date of birth because then we don’t need to update it every time it is their birthday. In fact, you can’t update it since we never kept track of their birthday.

Also, it is a bit strange to only include the boat’s length as an attribute in the boats table. If it were to be expanded, we could create a new table ‘boat-specifications’ with primary key bid and also cascade deletes from the main boat table. Here we could go more extensive on the specifications of the boat and remove it from the main boat table.
It may also be worthwhile for the company to keep track of an eid associated with a reservation to keep track of each employee's sales. This also entails creating an employee table where we can store their information.

If we have an employee table, we can also create a compensation table for their annual salary. Additionally, it would also be useful to add a department table to keep track of what departments employees are in. The final database can be quickly expanded to this:



Due to lack of time, I only implemented the repairs table and did the test queries on that.