Andy Jeong
Professor Sokolov
ECE 464: Databases
October 29, 2019
Problem Set 1

Part 1 (see “pset1_1.sql”)

1. Select, for each boat, the sailor who made the highest number of reservations for that boat.
(fixed)

```
mysql> SELECT DISTINCT b.bid, s.sname, count(*) as rank
    ->
    -> FROM boats b join reserves r on b.bid = r.bid
    ->
    -> JOIN sailors s on s.sid = r.sid
    ->
    -> GROUP BY b.bid, b.bname, s.sid, s.sname
    ->
    -> HAVING count(*) >= all(
    ->
    ->                        SELECT count(*)
    ->
    ->                        FROM reserves r1
    ->
    ->                        WHERE r1.bid = b.bid
    ->
    ->                        GROUP BY r1.sid
    ->
    ->                      )
    ->
    -> ORDER BY b.bid, s.sname;
```

```
+-----+----------+------+
| bid | sname    | rank |
+-----+----------+------+
| 101 | dusting  |    1 |
| 101 | horatio  |    1 |
| 102 | dusting  |    1 |
| 102 | horatio  |    1 |
| 102 | lubber   |    1 |
| 103 | dusting  |    1 |
| 103 | horatio  |    1 |
| 103 | lubber   |    1 |
| 104 | dusting  |    1 |
| 104 | emilio   |    1 |
| 104 | figaro   |    1 |
| 104 | lubber   |    1 |
| 104 | scruntus |    1 |
| 105 | emilio   |    1 |
| 105 | figaro   |    1 |
| 105 | stum     |    1 |
| 106 | jit      |    2 |
| 107 | dan      |    1 |
| 108 | dye      |    1 |
| 109 | dye      |    1 |
| 109 | jit      |    1 |
| 109 | stum     |    1 |
| 109 | vin      |    1 |
| 110 | dan      |    2 |
| 111 | dan      |    1 |
| 112 | ossola   |    1 |
+-----+----------+------+
26 rows in set (0.00 sec)
```

2. boats that have never been reserved (list the id and the name).

```
mysql> SELECT b.bid, b.bname, count(r.bid) as reserve_cnt
    ->
    -> FROM boats b join reserves r
    ->
    -> ON b.bid = r.bid
    ->
    -> GROUP BY r.bid;
+-----+-----------+-------------+
| bid | bname     | reserve_cnt |
+-----+-----------+-------------+
| 101 | Interlake |           2 |
| 102 | Interlake |           3 |
| 103 | Clipper   |           3 |
| 104 | Clipper   |           5 |
| 105 | Marine    |           3 |
| 106 | Marine    |           3 |
| 107 | Marine    |           1 |
| 108 | Driftwood |           1 |
| 109 | Driftwood |           4 |
| 110 | Klapser   |           3 |
| 111 | Sooney    |           1 |
| 112 | Sooney    |           1 |
+-----+-----------+-------------+
12 rows in set (0.00 sec)
```

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

```
mysql> SELECT s.sid, s.sname, r.bid
    ->
    -> FROM sailors s join reserves r
    ->
    -> ON s.sid  = r.sid
    ->
    -> WHERE r.bid = ALL (
    ->
    ->     SELECT b.bid FROM boats b
    ->
    ->     WHERE b.color = 'red'
    ->
    -> );
Empty set (0.00 sec)
```

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

```
mysql> SELECT DISTINCT s.sid, s.sname
    ->
    -> FROM sailors as s
    ->
    -> WHERE 'red'= ALL (
    ->
    ->     SELECT b.color FROM boats b join reserves r
    ->
    ->     ON b.bid = r.bid where r.sid = s.sid
    ->
    -> ) AND s.sid IN (
    ->
    ->     SELECT r.sid FROM reserves r
    ->
    -> );
+-----+----------+
| sid | sname    |
+-----+----------+
|  23 | emilio   |
|  24 | scruntus |
|  35 | figaro   |
|  61 | ossola   |
|  62 | shaun    |
+-----+----------+
5 rows in set (0.01 sec)
```

5. For which boat are there the most reservations?

```
mysql> SELECT b.bid, b.bname, count(*)
    ->
    -> FROM boats as b join reserves as r
    ->
    -> ON b.bid = r.bid
    ->
    -> GROUP BY b.bid, b.bname
    ->
    -> ORDER BY count(*) DESC
    ->
    -> LIMIT 1;
+-----+---------+----------+
| bid | bname   | count(*) |
+-----+---------+----------+
| 104 | Clipper |        5 |
+-----+---------+----------+
1 row in set (0.00 sec)
```

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

```
mysql> SELECT s.sid, s.sname
    ->
    -> FROM sailors as s
    ->
    -> LEFT JOIN (
    ->
    ->     reserves as r
    ->
    ->     INNER JOIN boats as b
    ->
    ->     ON r.bid = b.bid AND b.color='red'
    ->
    ->     )
    ->
    -> ON s.sid = r.sid
    ->
    -> WHERE b.color IS NULL;
+-----+---------+
| 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)
```

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

```
mysql> select avg(s.age) from sailors s where s.rating = 10 group by age;
+------------+
| avg(s.age) |
+------------+
|    35.0000 |
+------------+
1 row in set (0.01 sec)
```

Part 2 (see “pset1_2.py”, “pset1_2_test.py”)

Represent the sailors and boats schema using an ORM - I prefer SQLAlchemy but students have the freedom to choose their own language and ORM. Show that it is fully functional by writing tests using the data from part 1 (writing the queries for the questions in Part 1) - I prefer pytest but students are have the freedom to choose their own testing framework.

Part 3 (see “pset1_3.py”)

Students are hired as software consults for a small business boat rental that is experiencing a heavy influx of tourism in its area. This increase is hindering operations of the mom/pop shop that uses paper/pen for most tasks. Students should explore "inefficient processes" the business may have and propose ideas for improvements - in the form of a brief write-up. Expand the codebase from part 2 to include a few jobs, reports, integrity checks, and/or other processes that would be beneficial to the business. Use the data provided in part 1 and expand it to conduct tests and show functionality. Examples include, but are not limited to:

Bi weekly payment query, Monthly accounting manager, Daily inventory control, Inventory repair tracker (and cost analysis)

Some inefficient states for the current boat rental business might include:
1) manual management of boat payments
2) manual management of monthly boat checkup/repair history
3) manual counting of currently available boats for checkup, other personal uses

Some possible points of improvement include:
1) store payment history for each boat (keep track of outstanding receivables, whether rental fee has been paid)
2) keep track of checkup history for each boat
3) store current available states of each boat

To store such data, one can represent the schema as follows:

1. PaymentHistory
- primary key: sid
- sid (foreign key to Sailor.sid) :: Integer
- chargeDate :: DateTime
- daysTillDue :: Integer
- paid :: Boolean

2. CheckupHistory
- primary key: (bid, lastcheckdate)
- bid (foreign key to Boats.bid) :: Integer
- lastcheckdate :: DateTime
- problemDetecte :: Integer (0 if none, 1 if any, 2, 3... if other)

3. CurrentlyAvailable
- primary key: bid
- bid (foreign key to Boats.bid) :: Integer
- available :: Boolean

"Extra Credit"
Use a web code review platform so I can write comments for review. I should get a link to a review platform and be able to easily write comments - perhaps after linking with github or creating a free account. Ones that I have found good are codacy.com and reviewable.io. This will help prepare you for the final project and is highly recommended.