DATABASES
PSET 1

Derek Lee
Professor Sokolov
ECE-464

October 13, 2021

Part 1

Download the dataset and schema of sailors and boats from our in class discussion. Write SQL queries to answer the following questions. Include your query (and its output from your terminal in a presentable fashion) in your submissions.

Question 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 bid, bname, counts
FROM boats
INNER JOIN (
    SELECT bid, COUNT(*) AS counts
    FROM reserves
    GROUP BY bid
    HAVING counts > 0
) AS temp1
USING(bid);
```

```
+-----+-----------+--------+
| bid | bname     | counts |
+-----+-----------+--------+
| 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 |
+-----+-----------+--------+
```

```
mysql> SELECT bid, bname, counts
    -> FROM boats
    -> INNER JOIN (
    ->     SELECT bid, COUNT(*) AS counts
    ->     FROM reserves
    ->     GROUP BY bid
    ->     HAVING counts > 0
    -> ) AS temp1
    -> USING(bid);
+-----+-----------+--------+
| bid | bname     | counts |
+-----+-----------+--------+
| 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 |
+-----+-----------+--------+
```

Figure 1: Results to Question 1

Question 2

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

```
SELECT sid, sname
FROM (
    SELECT sid, sname, COUNT(*) AS counts
    FROM (
      SELECT DISTINCT sid, sname, bid
      FROM boats
      INNER JOIN (
        SELECT *
        FROM sailors
        INNER JOIN reserves
        USING(sid)
      ) AS temp1
```

```
      USING(bid)
      WHERE color = 'red'
    ) AS temp2
    GROUP BY sid, sname
) AS temp3
INNER JOIN (
  SELECT COUNT(DISTINCT bid) AS max_count
  FROM boats
  WHERE color = 'red'
) AS temp4
ON counts = max_count;

+-----------+
| Empty set |
+-----------+
```

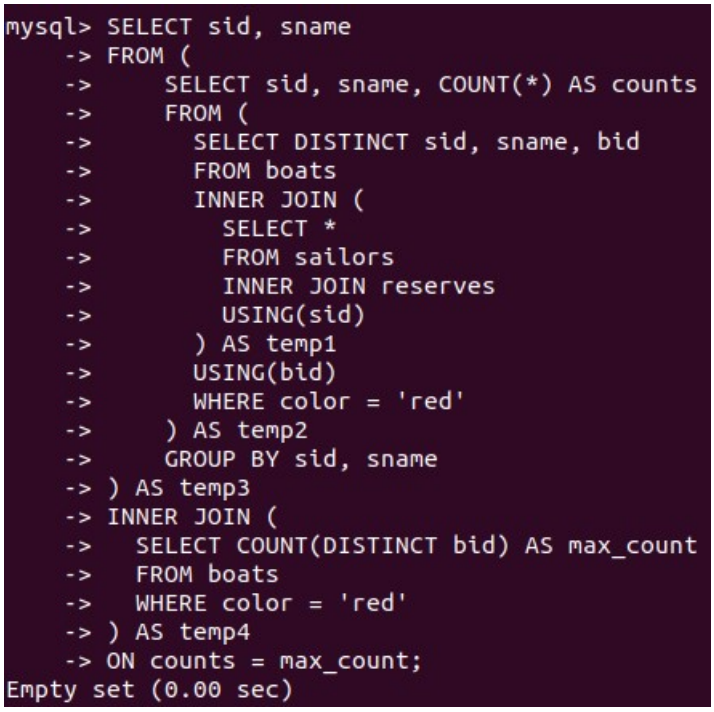

Figure 2: Results to Question 2

Question 3

List those sailors who have reserved only red boats.

```
SELECT sid, sname
FROM sailors
INNER JOIN (
    SELECT DISTINCT sid
    FROM reserves
    WHERE sid NOT IN (
        SELECT sid
        FROM reserves
        INNER JOIN (
            SELECT bid
            FROM boats
            WHERE color != 'red'
        ) AS temp1
        USING(bid)
    )
) AS temp2
USING(sid)
ORDER BY sid;
```

```
+-----+----------+
| sid | sname    |
+-----+----------+
|  23 | emilio   |
|  24 | scruntus |
|  35 | figaro   |
|  61 | ossola   |
|  62 | shaun    |
+-----+----------+
```

```
mysql> SELECT sid, sname
    -> FROM sailors
    -> INNER JOIN (
    ->     SELECT sid
    ->     FROM reserves
    ->     WHERE sid NOT IN (
    ->         SELECT sid
    ->         FROM reserves
    ->         INNER JOIN (
    ->             SELECT bid
    ->             FROM boats
    ->             WHERE color != 'red'
    ->         ) AS temp1
    ->         USING(bid)
    ->     )
    -> ) AS temp2
    -> USING(sid);
+-----+----------+
| sid | sname    |
+-----+----------+
|  23 | emilio   |
|  23 | emilio   |
|  24 | scruntus |
|  35 | figaro   |
|  35 | figaro   |
|  61 | ossola   |
|  62 | shaun    |
+-----+----------+
```

Figure 3: Results to Question 3

Question 4

For which boat are there the most reservations?

```
SELECT bid, bname
FROM boats
INNER JOIN (
    SELECT bid
    FROM (
        SELECT bid, COUNT(*) AS counts
        FROM reserves
        GROUP BY bid
    ) AS temp1
    WHERE counts = (
```

```
        SELECT MAX(counts)
        FROM (
            SELECT COUNT(*) AS counts
            FROM reserves
            GROUP BY bid
        ) AS temp2
    )
) AS temp3
USING(bid);

+-----+-----------+
| bid | bname     |
+-----+-----------+
| 104 | Clipper   |
+-----+-----------+
```

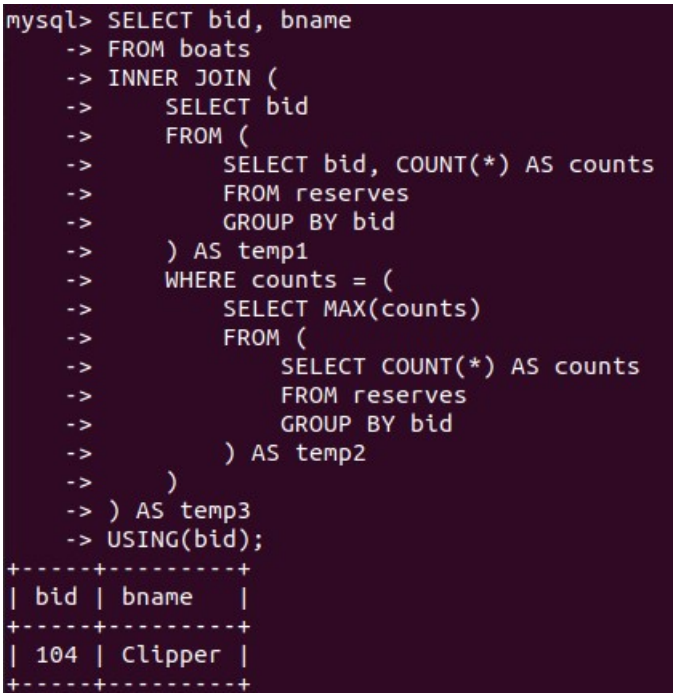

Figure 4: Results to Question 4

Question 5

Select all sailors who have never reserved a red boat.

```
SELECT sid, sname
FROM sailors
WHERE (sid, sname) NOT IN (
    SELECT sid, sname
    FROM sailors
    INNER JOIN (
        SELECT sid
        FROM reserves
        INNER JOIN (
            SELECT bid
            FROM boats
            WHERE color = 'red'
        ) AS temp1
        USING(bid)
    ) AS temp2
    USING(sid)
);
```

```
+-----+---------+
| sid | sname   |
+-----+---------+
|  29 | brutus  |
|  32 | andy    |
|  58 | rusty   |
|  60 | jit     |
|  71 | zorba   |
|  74 | horatio |
|  85 | art     |
|  90 | vin     |
|  95 | bob     |
+-----+---------+
```

```
mysql> SELECT sid, sname
    -> FROM sailors
    -> WHERE (sid, sname) NOT IN (
    ->     SELECT sid, sname
    ->     FROM sailors
    ->     INNER JOIN (
    ->         SELECT sid
    ->         FROM reserves
    ->         INNER JOIN (
    ->             SELECT bid
    ->             FROM boats
    ->             WHERE color = 'red'
    ->         ) AS temp1
    ->         USING(bid)
    ->     ) AS temp2
    ->     USING(sid)
    -> );
+-----+---------+
| sid | sname   |
+-----+---------+
|  29 | brutus  |
|  32 | andy    |
|  58 | rusty   |
|  60 | jit     |
|  71 | zorba   |
|  74 | horatio |
|  85 | art     |
|  90 | vin     |
|  95 | bob     |
+-----+---------+
```

Figure 5: Results to Question 5

Question 6

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

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

```
+----------+
| AVG(age) |
+----------+
|  35.0000 |
+----------+
```

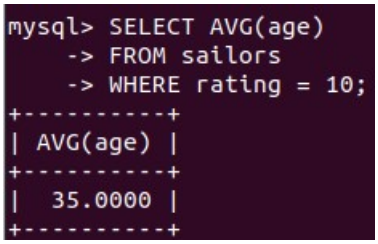

Figure 6: Results to Question 6

Question 7

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

```
SELECT sailors.rating, sid, sname, age
FROM sailors
INNER JOIN (
    SELECT rating, MIN(age) AS min_age
    FROM sailors
    GROUP BY rating
) AS temp1
ON sailors.rating = temp1.rating
AND age = min_age
ORDER BY rating, sid;
```

```
+--------+-----+----------+------+
| 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 |
+--------+-----+----------+------+
```

```
mysql> SELECT sailors.rating, sid, sname, age
    -> FROM sailors
    -> INNER JOIN (
    ->     SELECT rating, MIN(age) AS min_age
    ->     FROM sailors
    ->     GROUP BY rating
    -> ) AS temp1
    -> ON sailors.rating = temp1.rating
    -> AND age = min_age
    -> ORDER BY rating, sid;
+--------+-----+----------+------+
| 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 |
+--------+-----+----------+------+
```

Figure 7: Results to Question 7

Question 8

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

```
SELECT bid, sid, sname
FROM sailors
INNER JOIN (
```

```
    SELECT bid, sid, COUNT(*) AS counts, max_counts
    FROM reserves
    INNER JOIN (
        SELECT bid, MAX(counts) AS max_counts
        FROM (
         SELECT bid, sid, COUNT(*) AS counts
         FROM reserves
         GROUP BY bid, sid
        ) AS temp1
        GROUP BY bid
    ) AS temp2
    USING(bid)
    GROUP BY bid, sid
) AS temp3
USING(sid)
WHERE counts = max_counts
ORDER BY bid, sid;

+-----+-----+----------+
| bid | sid | sname    |
+-----+-----+----------+
| 101 |  22 | dusting  |
| 101 |  64 | horatio  |
| 102 |  22 | dusting  |
| 102 |  31 | lubber   |
| 102 |  64 | horatio  |
| 103 |  22 | dusting  |
| 103 |  31 | lubber   |
| 103 |  74 | horatio  |
| 104 |  22 | dusting  |
| 104 |  23 | emilio   |
| 104 |  24 | scruntus |
| 104 |  31 | lubber   |
| 104 |  35 | figaro   |
| 105 |  23 | emilio   |
| 105 |  35 | figaro   |
| 105 |  59 | stum     |
| 106 |  60 | jit      |
```

```
| 107 |  88 | dan      |
| 108 |  89 | dye      |
| 109 |  59 | stum     |
| 109 |  60 | jit      |
| 109 |  89 | dye      |
| 109 |  90 | vin      |
| 110 |  88 | dan      |
| 111 |  88 | dan      |
| 112 |  61 | ossola   |
+-----+-----+----------+
```

```
mysql> SELECT bid, sid, sname
    -> FROM sailors
    -> INNER JOIN (
    ->     SELECT bid, sid, COUNT(*) AS counts, max_counts
    ->     FROM reserves
    ->     INNER JOIN (
    ->         SELECT bid, MAX(counts) AS max_counts
    ->         FROM (
    ->         SELECT bid, sid, COUNT(*) AS counts
    ->         FROM reserves
    ->         GROUP BY bid, sid
    ->         ) AS temp1
    ->         GROUP BY bid
    ->     ) AS temp2
    ->     USING(bid)
    ->     GROUP BY bid, sid
    -> ) AS temp3
    -> USING(sid)
    -> WHERE counts = max_counts
    -> ORDER BY bid, sid;
+-----+-----+----------+
| bid | sid | sname    |
+-----+-----+----------+
| 101 |  22 | dusting  |
| 101 |  64 | horatio  |
| 102 |  22 | dusting  |
| 102 |  31 | lubber   |
| 102 |  64 | horatio  |
| 103 |  22 | dusting  |
| 103 |  31 | lubber   |
| 103 |  74 | horatio  |
| 104 |  22 | dusting  |
| 104 |  23 | emilio   |
| 104 |  24 | scruntus |
| 104 |  31 | lubber   |
| 104 |  35 | figaro   |
| 105 |  23 | emilio   |
| 105 |  35 | figaro   |
| 105 |  59 | stum     |
| 106 |  60 | jit      |
| 107 |  88 | dan      |
| 108 |  89 | dye      |
| 109 |  59 | stum     |
| 109 |  60 | jit      |
| 109 |  89 | dye      |
| 109 |  90 | vin      |
| 110 |  88 | dan      |
| 111 |  88 | dan      |
| 112 |  61 | ossola   |
+-----+-----+----------+
```

Figure 8: Results to Question 8

Alternative query:

```
WITH T AS (
    SELECT bid, sid, COUNT(*) AS counts
    FROM reserves
    GROUP BY bid, sid
)
SELECT bid, sid, sname
FROM sailors
INNER JOIN (
    SELECT bid, sid
    FROM T
    INNER JOIN (
        SELECT bid, MAX(counts) AS max_counts
        FROM T
        GROUP BY bid
    ) AS temp1
    USING(bid)
    WHERE counts = max_counts
) AS temp2
USING(sid)
ORDER BY bid, sid
```

Part 2

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 with a testing framework 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.

Code attached at the end of the document.

Part 3

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)*

I added a Reviews table to store customer reviews for their reservations. I have also modified the Reserves table to include a rsrvid to use with the new Reviews table. The Reviews table will store the rsrvid, the content of the review, a rating, and the date of the review. The test data I created has ratings from 1-5 (e.g. 1-5 stars), but it could theoretically be from any range, like 1-10, 1-100, etc.

Some example use cases are to find 1-star reviews and see what the comments are. Another use case is to find the average rating for all reviews. This can also be done for each boat color, so we would have information on the average rating for red, green, and blue boats.

Code attached at the end of the document.

Code

part1.sql

```
1  USE pset1;
2
3  /* Q1 */
4  SELECT bid, bname, counts
5  FROM boats
6  INNER JOIN (
7      SELECT bid, COUNT(*) AS counts
```

```sql
    FROM reserves
    GROUP BY bid
    HAVING counts > 0
) AS temp1
USING(bid);

/* Q2 */
SELECT sid, sname
FROM (
    SELECT sid, sname, COUNT(*) AS counts
    FROM (
      SELECT DISTINCT sid, sname, bid
      FROM boats
      INNER JOIN (
        SELECT *
        FROM sailors
        INNER JOIN reserves
        USING(sid)
      ) AS temp1
      USING(bid)
      WHERE color = 'red'
    ) AS temp2
    GROUP BY sid, sname
) AS temp3
INNER JOIN (
  SELECT COUNT(DISTINCT bid) AS max_count
  FROM boats
  WHERE color = 'red'
) AS temp4
ON counts = max_count;

/* Q3 */
SELECT sid, sname
FROM sailors
INNER JOIN (
    SELECT DISTINCT sid
    FROM reserves
    WHERE sid NOT IN (
```

```
        SELECT sid
        FROM reserves
        INNER JOIN (
            SELECT bid
            FROM boats
            WHERE color != 'red'
        ) AS temp1
        USING(bid)
    )
) AS temp2
USING(sid)
ORDER BY sid;

/* Q4 */
SELECT bid, bname
FROM boats
INNER JOIN (
    SELECT bid
    FROM (
        SELECT bid, COUNT(*) AS counts
        FROM reserves
        GROUP BY bid
    ) AS temp1
    WHERE counts = (
        SELECT MAX(counts)
        FROM (
            SELECT COUNT(*) AS counts
            FROM reserves
            GROUP BY bid
        ) AS temp2
    )
) AS temp3
USING(bid);

/* Q5 */
SELECT sid, sname
FROM sailors
WHERE (sid, sname) NOT IN (
```

```
    SELECT sid, sname
    FROM sailors
    INNER JOIN (
        SELECT sid
        FROM reserves
        INNER JOIN (
            SELECT bid
            FROM boats
            WHERE color = 'red'
        ) AS temp1
        USING(bid)
    ) AS temp2
    USING(sid)
);

/* Q6 */
SELECT AVG(age)
FROM sailors
WHERE rating = 10;

/* Q7 */
SELECT sailors.rating, sid, sname, age
FROM sailors
INNER JOIN (
    SELECT rating, MIN(age) AS min_age
    FROM sailors
    GROUP BY rating
) AS temp1
ON sailors.rating = temp1.rating
AND age = min_age
ORDER BY rating, sid;

/* Q8 */
SELECT bid, sid, sname
FROM sailors
INNER JOIN (
    SELECT bid, sid, COUNT(*) AS counts, max_counts
    FROM reserves
```

```
122     INNER JOIN (
123         SELECT bid, MAX(counts) AS max_counts
124         FROM (
125                 SELECT bid, sid, COUNT(*) AS counts
126                 FROM reserves
127                 GROUP BY bid, sid
128         ) AS temp1
129         GROUP BY bid
130     ) AS temp2
131     USING(bid)
132     GROUP BY bid, sid
133 ) AS temp3
134 USING(sid)
135 WHERE counts = max_counts
136 ORDER BY bid, sid;
137 
138 /* Q8 Alternative */
139 WITH T AS (
140     SELECT bid, sid, COUNT(*) AS counts
141     FROM reserves
142     GROUP BY bid, sid
143 )
144 SELECT bid, sid, sname
145 FROM sailors
146 INNER JOIN (
147     SELECT bid, sid
148     FROM T
149     INNER JOIN (
150         SELECT bid, MAX(counts) AS max_counts
151         FROM T
152         GROUP BY bid
153     ) AS temp1
154     USING(bid)
155     WHERE counts = max_counts
156 ) AS temp2
157 USING(sid)
158 ORDER BY bid, sid
```

data.py

```
sailors = [
    (22,'dusting',7,45),
    (23,'emilio',7,45),
    (24,'scruntus',1,33),
    (29,'brutus',1,33),
    (31,'lubber',8,55),
    (32,'andy',8,25),
    (35,'figaro',8,55),
    (58,'rusty',10,35),
    (59,'stum',8,25),
    (60,'jit',10,35),
    (61,'ossola',7,16),
    (62,'shaun',10,35),
    (64,'horatio',7,16),
    (71,'zorba',10,35),
    (74,'horatio',9,25),
    (85,'art',3,25),
    (88,'dan',9,25),
    (89,'dye',3,25),
    (90,'vin',3,63),
    (95,'bob',3,63),
]

boats = [
    (101,'Interlake','blue',45),
    (102,'Interlake','red',45),
    (103,'Clipper','green',40),
    (104,'Clipper','red',40),
    (105,'Marine','red',35),
    (106,'Marine','green',35),
    (107,'Marine','blue',35),
    (108,'Driftwood','red',35),
    (109,'Driftwood','blue',35),
    (110,'Klapser','red',30),
    (111,'Sooney','green',28),
    (112,'Sooney','red',28),
```

```
]

reserves = [
    (2,22,101,'1998/10/10'),
    (5,22,102,'1998/10/10'),
    (9,22,103,'1998/8/10'),
    (15,22,104,'1998/7/10'),
    (17,23,104,'1998/10/10'),
    (21,23,105,'1998/11/10'),
    (24,24,104,'1998/10/10'),
    (25,31,102,'1998/11/10'),
    (26,31,103,'1998/11/6'),
    (29,31,104,'1998/11/12'),
    (30,35,104,'1998/8/10'),
    (33,35,105,'1998/11/6'),
    (36,59,105,'1998/7/10'),
    (37,59,106,'1998/11/12'),
    (41,59,109,'1998/11/10'),
    (42,60,106,'1998/9/5'),
    (43,60,106,'1998/9/8'),
    (45,60,109,'1998/7/10'),
    (54,61,112,'1998/9/8'),
    (78,62,110,'1998/11/6'),
    (81,64,101,'1998/9/5'),
    (86,64,102,'1998/9/8'),
    (90,74,103,'1998/9/8'),
    (91,88,107,'1998/9/8'),
    (92,88,110,'1998/11/12'),
    (95,88,110,'1998/9/5'),
    (101,88,111,'1998/9/8'),
    (102,89,108,'1998/10/10'),
    (104,89,109,'1998/8/10'),
    (107,90,109,'1998/10/10'),
]

reviews = [
    (2,"",5,'1998/10/11'),
    (9,"great service",5,'1998/9/10'),
```

```
    (21,"I was charged the wrong amount and the owners refused to fix it.",1,'1998/12/15'),
    (25,"The boat was a little shabby, but overall the experience was good.",4.5,'1998/11/10'),
    (30,"Never coming back again.",1,'1998/8/10'),
    (41,"",2,'1998/11/10'),
    (42,"The cashier was rude.",2,'1998/9/6'),
    (54,"My boat had a leak in it and I lost all of my stuff in the lake!",1.5,'1998/9/8'),
    (90,"A great business in a great location! Will definitely come back.",5,'1998/9/8'),
]
```

sailors_orm.py

```
from typing import List, Tuple

from sqlalchemy import create_engine, Column, Integer, String, DateTime, PrimaryKeyConstraint, ForeignKey
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker, backref, relationship
import datetime

from sqlalchemy.sql.sqltypes import Numeric

from data import sailors, boats, reserves, reviews

# Used to get DB connection
import os, sys
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__)))) # Get parent dir
from dbInfo import Info

engine = create_engine('mysql+mysqlconnector://' + Info.connect + '/pset1', echo=True)

Session = sessionmaker(bind=engine)
session = Session()
```

```
21
22 Base = declarative_base()
23
24
25 class Sailor(Base):
26     __tablename__ = 'sailors'
27
28     sid = Column(Integer, primary_key=True)
29     sname = Column(String(20))
30     rating = Column(Integer)
31     age = Column(Integer)
32
33     def __init__(self, data: Tuple[Integer, String, Integer,
       ↪  String]):
34         self.sid = data[0]
35         self.sname = data[1]
36         self.rating = data[2]
37         self.age = data[3]
38
39     def __repr__(self):
40         return "<Sailor(id=%s, name='%s', rating=%s, age=%s)>"
           ↪  % (self.sid, self.sname, self.rating, self.age)
41
42
43 class Boat(Base):
44     __tablename__ = 'boats'
45
46     bid = Column(Integer, primary_key=True)
47     bname = Column(String(20))
48     color = Column(String(20))
49     length = Column(Integer)
50
51     reservations = relationship('Reservation',
52                                 backref=backref('boat',
                                    ↪  cascade='delete'))
53
54     def __init__(self, data: Tuple[Integer, String, String,
       ↪  Integer]):
```

```
55          self.bid = data[0]
56          self.bname = data[1]
57          self.color = data[2]
58          self.length = data[3]
59
60      def __repr__(self):
61          return "<Boat(id=%s, name='%s', color=%s)>" %
            ↪  (self.bid, self.bname, self.color)
62
63
64  class Reservation(Base):
65      __tablename__ = 'reserves'
66
67      rsrvid = Column(Integer, primary_key=True)
68      sid = Column(Integer, ForeignKey('sailors.sid'))
69      bid = Column(Integer, ForeignKey('boats.bid'))
70      day = Column(DateTime)
71
72      def __init__(self, data: Tuple[Integer, Integer, Integer,
        ↪  String]):
73          self.rsrvid = data[0]
74          self.sid = data[1]
75          self.bid = data[2]
76          self.day = datetime.datetime.strptime(data[3],
            ↪  "%Y/%m/%d")
77
78      def __repr__(self):
79          return "<Reservation(rsrvid=%s, sid=%s, bid=%s,
            ↪  day=%s)>" % (self.rsrvid, self.sid, self.bid,
            ↪  self.day)
80
81
82  class Review(Base):
83      __tablename__ = 'reviews'
84      __table_args__ = (PrimaryKeyConstraint('rsrvid',
        ↪  'contents', 'rating', 'day'), {})
85
86      rsrvid = Column(Integer, ForeignKey('reserves.rsrvid'))
```

```
87      contents = Column(String(160))
88      rating = Column(Numeric(2,1))
89      day = Column(DateTime)
90
91      def __init__(self, data: Tuple[Integer, String, Numeric,
        ↪  String]):
92          self.rsrvid = data[0]
93          self.contents = data[1]
94          self.rating = data[2]
95          self.day = datetime.datetime.strptime(data[3],
            ↪  "%Y/%m/%d")
96
97      def __repr__(self):
98          return "<Review(rsrvid=%s, contents=%s, rating=%s,
            ↪  day=%s)>" % (self.rsrvid, self.contents,
            ↪  self.rating, self.day)
99
100
101 def initTable(tables: List[Tuple[String, List]]):
102     # Reset the tables
103     [table[0].__table__.drop(engine, checkfirst=True) for table
        ↪  in tables]
104     [table[0].__table__.create(engine, checkfirst=True) for
        ↪  table in reversed(tables)]
105
106     # Insert data
107     # For each table, uses data to initalize specified table
        ↪  class and then inserts data
108     [session.bulk_save_objects([table[0](x) for x in table[1]])
109         for table in reversed(tables)]
110     session.commit()
111
112
113 # Drop, Create, Insert Tables
114 if __name__ == '__main__':
115     initTable([
116         (Review, reviews),
117         (Reservation, reserves),
```

```
        (Sailor, sailors),
        (Boat, boats),
    ])
    print("Succeeded")
```

test_sailors_orm.py

```
from sqlalchemy import create_engine, func
from sqlalchemy.orm import sessionmaker
from sqlalchemy.sql import select
from sqlalchemy.sql.expression import text, distinct
from sailors_orm import Sailor, Boat, Reservation, Review

# Used to get DB connection
import os, sys
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
 ↪  # Get parent dir
from dbInfo import Info

engine = create_engine('mysql+mysqlconnector://' + Info.connect
 ↪  + '/pset1', echo=True)

Session = sessionmaker(bind=engine)
session = Session()

def test_q1():
    ans = [
        (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),
```

```
30          (112, 'Sooney', 1),
31      ]
32
33      innerStatement = select(Reservation.bid,
        ↪  func.count().label("num_reserves")) \
34          .select_from(Reservation) \
35          .group_by(Reservation.bid) \
36          .having(text("num_reserves > 0")) \
37          .alias("temp1")
38      statement = select(Boat.bid, Boat.bname,
        ↪  text("num_reserves")) \
39          .select_from(Boat) \
40          .join(innerStatement)
41      results = session.execute(statement).fetchall()
42
43      assert results == ans
44
45  def test_q2():
46      ans = []
47
48      innerStatement1 = select(text("sailors.*"),
        ↪  Reservation.bid) \
49          .select_from(Sailor) \
50          .join(Reservation) \
51          .alias("temp1")
52      innerStatement2 = select(text("sid"), text("sname"),
        ↪  Boat.bid).distinct() \
53          .select_from(Boat) \
54          .join(innerStatement1, text("boats.bid = temp1.bid")) \
55          .where(Boat.color == "red") \
56          .alias("temp2")
57      innerStatement3 = select(text("sid"), text("sname"),
        ↪  func.count().label("counts")) \
58          .select_from(innerStatement2) \
59          .group_by(text("sid"), text("sname")) \
60          .alias("temp3")
```

```
61      innerStatement4 =
        ↪  select(func.count(distinct(Boat.bid)).label("max_count"))
        ↪  \
62          .select_from(Boat) \
63          .where(Boat.color == "red") \
64          .alias("temp4")
65      statement = select(text("sid"), text("sname")) \
66          .select_from(innerStatement3) \
67          .join(innerStatement4, text("counts = max_count"))
68      results = session.execute(statement).fetchall()
69
70      assert results == ans
71
72  def test_q3():
73      ans = [
74          (23, 'emilio'),
75          (24, 'scruntus'),
76          (35, 'figaro'),
77          (61, 'ossola'),
78          (62, 'shaun'),
79      ]
80
81      excludeInnerStatement = select(Boat.bid) \
82          .select_from(Boat) \
83          .where(Boat.color != "red") \
84          .alias("temp1")
85      excludeStatement = select(Reservation.sid) \
86          .select_from(Reservation) \
87          .join(excludeInnerStatement)
88      excludeStatement =
        ↪  excludeStatement.compile(compile_kwargs={"literal_binds":
        ↪  True})
89      innerStatement = select(Reservation.sid).distinct() \
90          .select_from(Reservation) \
91          .where(text("sid NOT IN (" + str(excludeStatement) +
            ↪  ")")) \
92          .alias("temp2")
93      statement = select(Sailor.sid, Sailor.sname) \
```

```
94          .select_from(Sailor) \
95          .join(innerStatement) \
96          .order_by(Sailor.sid)
97      results = session.execute(statement).fetchall()
98
99      assert results == ans
100
101 def test_q4():
102     ans = [(104, 'Clipper')]
103
104     innerStatement1 = select(Reservation.bid,
        ↪  func.count().label("counts")) \
105         .select_from(Reservation) \
106         .group_by(Reservation.bid) \
107         .alias("temp1")
108     innerWhereStatement2 = select(func.count().label("counts"))
        ↪  \
109         .select_from(Reservation) \
110         .group_by(Reservation.bid) \
111         .alias("temp2")
112     innerStatement3 = select(text("bid AS bid_temp")) \
113         .select_from(innerStatement1) \
114         .where(text("counts = (" +
            ↪  str(select(func.max(text("counts"))) \
115             .select_from(innerWhereStatement2)) + ")")) \
116         .alias("temp3")
117     statement = select(Boat.bid, Boat.bname) \
118         .select_from(Boat) \
119         .join(innerStatement3, Boat.bid == text("bid_temp"))
120     results = session.execute(statement).fetchall()
121
122     assert results == ans
123
124 def test_q5():
125     ans = [
126         (29, 'brutus'),
127         (32, 'andy'),
128         (58, 'rusty'),
```

```
        (60, 'jit'),
        (71, 'zorba'),
        (74, 'horatio'),
        (85, 'art'),
        (90, 'vin'),
        (95, 'bob'),
    ]

    excludeInnerStatement2 = select(Boat.bid) \
        .select_from(Boat) \
        .where(Boat.color == "red") \
        .alias("temp1")
    excludeInnerStatement = select(Reservation.sid) \
        .select_from(Reservation) \
        .join(excludeInnerStatement2) \
        .alias("temp2")
    excludeStatement = select(Sailor.sid, Sailor.sname) \
        .select_from(Sailor) \
        .join(excludeInnerStatement)
    excludeStatement =
    ↪  excludeStatement.compile(compile_kwargs={"literal_binds":
    ↪  True})
    statement = select(Sailor.sid, Sailor.sname) \
        .select_from(Sailor) \
        .where(text("(sid, sname) NOT IN (" +
        ↪  str(excludeStatement) + ")"))
    results = session.execute(statement).fetchall()

    assert results == ans

def test_q6():
    ans = [(35,)]

    statement = select(func.avg(Sailor.age)) \
        .select_from(Sailor) \
        .where(Sailor.rating == 10)
    results = session.execute(statement).fetchall()

```

```
164     assert results == ans
165
166 def test_q7():
167     ans = [
168         (1, 24, 'scruntus', 33),
169         (1, 29, 'brutus', 33),
170         (3, 85, 'art', 25),
171         (3, 89, 'dye', 25),
172         (7, 61, 'ossola', 16),
173         (7, 64, 'horatio', 16),
174         (8, 32, 'andy', 25),
175         (8, 59, 'stum', 25),
176         (9, 74, 'horatio', 25),
177         (9, 88, 'dan', 25),
178         (10, 58, 'rusty', 35),
179         (10, 60, 'jit', 35),
180         (10, 62, 'shaun', 35),
181         (10, 71, 'zorba', 35),
182     ]
183
184     innerStatement = select(Sailor.rating,
        ↪ func.min(Sailor.age).label("min_age")) \
185         .select_from(Sailor) \
186         .group_by(Sailor.rating) \
187         .alias("temp1")
188     statement = select(Sailor.rating, Sailor.sid, Sailor.sname,
        ↪ Sailor.age) \
189         .select_from(Sailor) \
190         .join(innerStatement, text("sailors.rating =
            ↪ temp1.rating AND age = min_age")) \
191         .order_by(Sailor.rating, Sailor.sid)
192     results = session.execute(statement).fetchall()
193
194     assert results == ans
195
196 def test_q8():
197     ans = [
198         (101, 22, 'dusting'),
```

```
        (101, 64, 'horatio'),
        (102, 22, 'dusting'),
        (102, 31, 'lubber'),
        (102, 64, 'horatio'),
        (103, 22, 'dusting'),
        (103, 31, 'lubber'),
        (103, 74, 'horatio'),
        (104, 22, 'dusting'),
        (104, 23, 'emilio'),
        (104, 24, 'scruntus'),
        (104, 31, 'lubber'),
        (104, 35, 'figaro'),
        (105, 23, 'emilio'),
        (105, 35, 'figaro'),
        (105, 59, 'stum'),
        (106, 60, 'jit'),
        (107, 88, 'dan'),
        (108, 89, 'dye'),
        (109, 59, 'stum'),
        (109, 60, 'jit'),
        (109, 89, 'dye'),
        (109, 90, 'vin'),
        (110, 88, 'dan'),
        (111, 88, 'dan'),
        (112, 61, 'ossola'),
    ]

    innerStatement1 = select(Reservation.bid, Reservation.sid,
    ↪  func.count().label("counts")) \
        .select_from(Reservation) \
        .group_by(Reservation.bid, Reservation.sid) \
        .alias("temp1")
    innerStatement2 = select(text("bid AS bid_temp"),
    ↪  func.max(text("counts")).label("max_counts")) \
        .select_from(innerStatement1) \
        .group_by(text("bid_temp")) \
        .alias("temp2")
```

```
234     innerStatement3 = select(Reservation.bid, Reservation.sid,
        ↪  func.count().label("counts"), text("max_counts")) \
235         .select_from(Reservation) \
236         .join(innerStatement2, Reservation.bid ==
            ↪  text("bid_temp")) \
237         .group_by(Reservation.bid, Reservation.sid) \
238         .alias("temp3")
239     statement = select(text("bid"), Sailor.sid, Sailor.sname) \
240         .select_from(Sailor) \
241         .join(innerStatement3) \
242         .where(text("counts = max_counts")) \
243         .order_by(text("bid"), Sailor.sid)
244     results = session.execute(statement).fetchall()
245
246     assert results == ans
247
248 def test_average_rating():
249     ans = [(3,)]
250
251     statement = select(func.avg(Review.rating)) \
252         .select_from(Review)
253     results = session.execute(statement).fetchall()
254
255     assert results == ans
256
257 def test_get_one_star_reviews():
258     ans = [
259         ("I was charged the wrong amount and the owners refused
            ↪  to fix it.",),
260         ("Never coming back again.",),
261     ]
262
263     statement = select(Review.contents) \
264         .select_from(Review) \
265         .where(Review.rating == 1) \
266         .order_by(Review.rsrvid)
267     results = session.execute(statement).fetchall()
268
```

```
269     assert results == ans
270
271 def test_get_average_rating_per_boat_color():
272     ans = [
273         ('blue', 3.5),
274         ('green', 4),
275         ('red', 2)
276     ]
277
278     statement = select(Boat.color, func.avg(Review.rating)) \
279         .select_from(Review) \
280         .join(Reservation) \
281         .join(Boat) \
282         .group_by(Boat.color)
283     results = session.execute(statement).fetchall()
284
285     assert results == ans
```