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

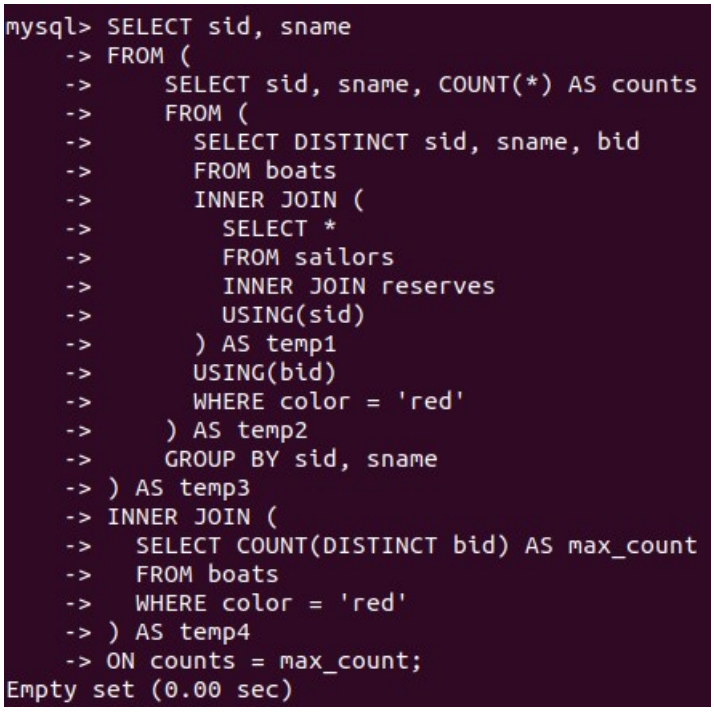

Figure 2: Results to Question 2

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

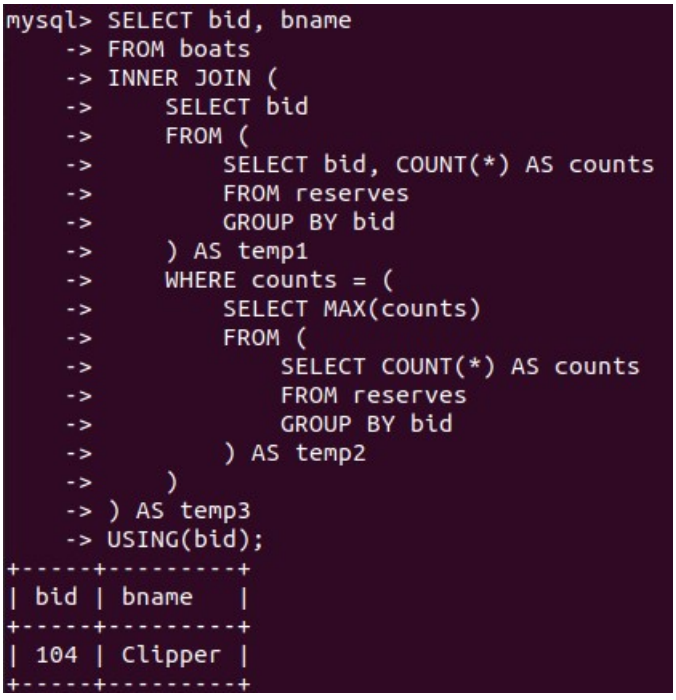

Figure 4: Results to Question 4

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

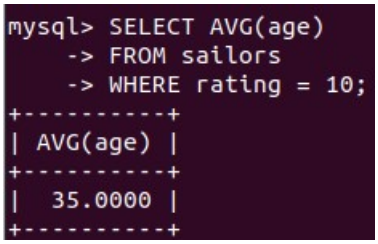

Figure 6: Results to Question 6

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

```
USE pset1;

/* Q1 */
SELECT bid, bname, counts
FROM boats
INNER JOIN (
```

```sql
    SELECT bid, COUNT(*) AS counts
    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 */
```

```sql
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
```

```sql
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;

/* Q8 Alternative */
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
```

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,'199
    (25,"The boat was a little shabby, but overall the experience was good.",4.5,
    (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,'1
    (90,"A great business in a great location! Will definitely come back.",5,'199
]
```

sailors_orm.py

```
from typing import List, Tuple

from sqlalchemy import create_engine, Column, Integer, String, DateTime, PrimaryK
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__)))) # Ge
from dbInfo import Info
```

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

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

Base = declarative_base()


class Sailor(Base):
    __tablename__ = 'sailors'

    sid = Column(Integer, primary_key=True)
    sname = Column(String(20))
    rating = Column(Integer)
    age = Column(Integer)

    def __init__(self, data: Tuple[Integer, String, Integer, String]):
        self.sid = data[0]
        self.sname = data[1]
        self.rating = data[2]
        self.age = data[3]

    def __repr__(self):
        return "<Sailor(id=%s, name='%s', rating=%s, age=%s)>" % (self.sid, self.


class Boat(Base):
    __tablename__ = 'boats'

    bid = Column(Integer, primary_key=True)
    bname = Column(String(20))
    color = Column(String(20))
    length = Column(Integer)

    reservations = relationship('Reservation',
                                backref=backref('boat', cascade='delete'))
```

```
    def __init__(self, data: Tuple[Integer, String, String, Integer]):
        self.bid = data[0]
        self.bname = data[1]
        self.color = data[2]
        self.length = data[3]

    def __repr__(self):
        return "<Boat(id=%s, name='%s', color=%s)>" % (self.bid, self.bname, self

class Reservation(Base):
    __tablename__ = 'reserves'

    rsrvid = Column(Integer, primary_key=True)
    sid = Column(Integer, ForeignKey('sailors.sid'))
    bid = Column(Integer, ForeignKey('boats.bid'))
    day = Column(DateTime)

    def __init__(self, data: Tuple[Integer, Integer, Integer, String]):
        self.rsrvid = data[0]
        self.sid = data[1]
        self.bid = data[2]
        self.day = datetime.datetime.strptime(data[3], "%Y/%m/%d")

    def __repr__(self):
        return "<Reservation(rsrvid=%s, sid=%s, bid=%s, day=%s)>" % (self.rsrvid,

class Review(Base):
    __tablename__ = 'reviews'
    __table_args__ = (PrimaryKeyConstraint('rsrvid', 'contents', 'rating', 'day'),

    rsrvid = Column(Integer, ForeignKey('reserves.rsrvid'))
    contents = Column(String(160))
    rating = Column(Numeric(2,1))
    day = Column(DateTime)
```

```
    def __init__(self, data: Tuple[Integer, String, Numeric, String]):
        self.rsrvid = data[0]
        self.contents = data[1]
        self.rating = data[2]
        self.day = datetime.datetime.strptime(data[3], "%Y/%m/%d")

    def __repr__(self):
        return "<Review(rsrvid=%s, contents=%s, rating=%s, day=%s)>" % (self.rsrv


def initTable(tables: List[Tuple[String, List]]):
    # Reset the tables
    [table[0].__table__.drop(engine, checkfirst=True) for table in tables]
    [table[0].__table__.create(engine, checkfirst=True) for table in reversed(tab

    # Insert data
    # For each table, uses data to initalize specified table class and then inser
    [session.bulk_save_objects([table[0](x) for x in table[1]])
        for table in reversed(tables)]
    session.commit()


# Drop, Create, Insert Tables
if __name__ == '__main__':
    initTable([
        (Review, reviews),
        (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__)))) # Ge
from dbInfo import Info

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

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),
        (112, 'Sooney', 1),
    ]

    innerStatement = select(Reservation.bid, func.count().label("num_reserves"))
        .select_from(Reservation) \
        .group_by(Reservation.bid) \
```

```
        .having(text("num_reserves > 0")) \
        .alias("temp1")
    statement = select(Boat.bid, Boat.bname, text("num_reserves")) \
        .select_from(Boat) \
        .join(innerStatement)
    results = session.execute(statement).fetchall()

    assert results == ans

def test_q2():
    ans = []

    innerStatement1 = select(text("sailors.*"), Reservation.bid) \
        .select_from(Sailor) \
        .join(Reservation) \
        .alias("temp1")
    innerStatement2 = select(text("sid"), text("sname"), Boat.bid).distinct() \
        .select_from(Boat) \
        .join(innerStatement1, text("boats.bid = temp1.bid")) \
        .where(Boat.color == "red") \
        .alias("temp2")
    innerStatement3 = select(text("sid"), text("sname"), func.count().label("coun
        .select_from(innerStatement2) \
        .group_by(text("sid"), text("sname")) \
        .alias("temp3")
    innerStatement4 = select(func.count(distinct(Boat.bid)).label("max_count")) \
        .select_from(Boat) \
        .where(Boat.color == "red") \
        .alias("temp4")
    statement = select(text("sid"), text("sname")) \
        .select_from(innerStatement3) \
        .join(innerStatement4, text("counts = max_count"))
    results = session.execute(statement).fetchall()

    assert results == ans

def test_q3():
```

```
    ans = [
        (23, 'emilio'),
        (24, 'scruntus'),
        (35, 'figaro'),
        (61, 'ossola'),
        (62, 'shaun'),
    ]

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

    assert results == ans

def test_q4():
    ans = [(104, 'Clipper')]

    innerStatement1 = select(Reservation.bid, func.count().label("counts")) \
        .select_from(Reservation) \
        .group_by(Reservation.bid) \
        .alias("temp1")
    innerWhereStatement2 = select(func.count().label("counts")) \
        .select_from(Reservation) \
```

```
        .group_by(Reservation.bid) \
        .alias("temp2")
    innerStatement3 = select(text("bid AS bid_temp")) \
        .select_from(innerStatement1) \
        .where(text("counts = (" + str(select(func.max(text("counts"))) \
            .select_from(innerWhereStatement2)) + ")")) \
        .alias("temp3")
    statement = select(Boat.bid, Boat.bname) \
        .select_from(Boat) \
        .join(innerStatement3, Boat.bid == text("bid_temp"))
    results = session.execute(statement).fetchall()

    assert results == ans

def test_q5():
    ans = [
        (29, 'brutus'),
        (32, 'andy'),
        (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":
    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()

    assert results == ans

def test_q7():
    ans = [
        (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),
    ]
```

```
    innerStatement = select(Sailor.rating, func.min(Sailor.age).label("min_age"))
        .select_from(Sailor) \
        .group_by(Sailor.rating) \
        .alias("temp1")
    statement = select(Sailor.rating, Sailor.sid, Sailor.sname, Sailor.age) \
        .select_from(Sailor) \
        .join(innerStatement, text("sailors.rating = temp1.rating AND age = min_a
        .order_by(Sailor.rating, Sailor.sid)
    results = session.execute(statement).fetchall()

    assert results == ans

def test_q8():
    ans = [
        (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
        .select_from(Reservation) \
        .group_by(Reservation.bid, Reservation.sid) \
        .alias("temp1")
    innerStatement2 = select(text("bid AS bid_temp"), func.max(text("counts")).la
        .select_from(innerStatement1) \
        .group_by(text("bid_temp")) \
        .alias("temp2")
    innerStatement3 = select(Reservation.bid, Reservation.sid, func.count().label
        .select_from(Reservation) \
        .join(innerStatement2, Reservation.bid == text("bid_temp")) \
        .group_by(Reservation.bid, Reservation.sid) \
        .alias("temp3")
    statement = select(text("bid"), Sailor.sid, Sailor.sname) \
        .select_from(Sailor) \
        .join(innerStatement3) \
        .where(text("counts = max_counts")) \
        .order_by(text("bid"), Sailor.sid)
    results = session.execute(statement).fetchall()

    assert results == ans

def test_average_rating():
    ans = [(3,)]

    statement = select(func.avg(Review.rating)) \
        .select_from(Review)
    results = session.execute(statement).fetchall()

    assert results == ans

def test_get_one_star_reviews():
```

```
    ans = [
        ("I was charged the wrong amount and the owners refused to fix it.",),
        ("Never coming back again.",),
    ]

    statement = select(Review.contents) \
        .select_from(Review) \
        .where(Review.rating == 1) \
        .order_by(Review.rsrvid)
    results = session.execute(statement).fetchall()

    assert results == ans

def test_get_average_rating_per_boat_color():
    ans = [
        ('blue', 3.5),
        ('green', 4),
        ('red', 2)
    ]

    statement = select(Boat.color, func.avg(Review.rating)) \
        .select_from(Review) \
        .join(Reservation) \
        .join(Boat) \
        .group_by(Boat.color)
    results = session.execute(statement).fetchall()

    assert results == ans
```