1.1
Query:

```
SELECT boats.bid, boats.bname, COUNT(boats.bid)
        FROM reserves JOIN boats ON boats.bid=reserves.bid
        GROUP BY reserves.bid, boats.bname, boats.bid;
```

Result:

bid	bname	count

List those sailors who have reserved only red boats.
For which boat are there the most reservations?
Select all sailors who have never reserved a red boat.
Find the average age of sailors with a rating of 10.
For each rating, find the name and id of the youngest sailor.
Select, for each boat, the sailor who made the highest number of reservations for that boat.

105	Marine	3
104	Clipper	5
112	Sooney	1
103	Clipper	3
101	Interlake	2
102	Interlake	3
108	Driftwood	1
109	Driftwood	4
110	Klapser	3
107	Marine	1
111	Sooney	1
106	Marine	3

1.2
Query:

```
select sailors.sname, sailors.sid from reserves
        inner join boats
                on boats.bid = reserves.bid
        inner join sailors
                on sailors.sid = reserves.sid
        where boats.color = 'red'
        group by sailors.sid
        having COUNT(boats.bid) = (select distinct COUNT(boats.bid) from boats
 where boats.color = 'red');
```

Result:

sname	sid

1.3
Query:

```
    select sailors.sname, sailors.sid from reserves r
        inner join boats b
            on b.bid=r.bid
        inner join sailors
            on sailors.sid = r.sid
        where b.color = 'red' and sailors.sid not in (select r.sid from reserv
es r inner join boats b on b.bid=r.bid where b.color != 'red')
        group by sailors.sid
```

Result:

sname	sid
emilio	23
scruntus	24
figaro	35
ossola	61
shaun	62

1.4
Query:

```
select reserves.bid, COUNT(reserves.bid) from reserves group by reserves.bid o
rder by COUNT(reserves.bid) desc limit 1
```

Result:

```
     |bid|count|
     |---|-----|
     |104|5    |


1.5
Query:
    select sailors.sname, sailors.sid from sailors where
        sailors.sid not in (select r.sid from reserves r inner join boats b on b.bid = r.bid where b.color = 'red')
Result:
     |sname  |sid|
     |-------|---|
     |brutus |29 |
     |andy   |32 |
     |rusty  |58 |
     |zorba  |71 |
     |horatio|74 |
     |art    |85 |
     |bob    |95 |
     |jit    |60 |
     |vin    |90 |
     |joe    |99 |

1.6
Query:
select AVG(sailors.age) from sailors where
        sailors.rating = 10
Result:
     |avg|
     |---|
     |35 |

--This doesn't break ties
--1.7
select s.sname, s.age, s.rating from sailors s
where age = (select MIN(age) from sailors s2 where s.rating = s2.rating) group by s.rating, s.sname, s.age;

--1.8
select sailors.sname, sailors.sid, COUNT(*) as reservationcount from reserves r
        join sailors
        on sailors.sid = r.sid, reservationcount = r.reservationcount
        group by sailors.sname, sailors.sid
        where reservationcount = (select COUNT(reserves.sid) as reservationcount, reserves.bid from reserves group by reserves.bid, reserves.sid);
```

```python
from orm import *
from sqlalchemy import select, func, distinct, desc
from sqlalchemy.orm import Session
import pytest

# Helper function


def cmp(orm_result, sql_result):
    assert len(orm_result) == len(
        sql_result
    ), f"orm result was not the same size as sql_result"
    for entry in sql_result:
        # print(entry)
        el = entry in orm_result
        assert (
            el
        ), f"Element Error: \n {entry} was present in orm return \n not present in the raw sql query"


def test_one():
    data_sel = select(Boat.bid, Boat.bname, func.count(Boat.bid))
    table_join = data_sel.join_from(Reservation, Boat)
    stmt = table_join.group_by(Reservation.bid, Boat.bid, Boat.bname)

    orm_result = conn.execute(stmt.order_by(Boat.bid)).fetchall()
    txt = """SELECT boats.bid, boats.bname, COUNT(boats.bid)
    FROM reserves JOIN boats ON boats.bid=reserves.bid
    GROUP BY reserves.bid, boats.bname, boats.bid"""
    sql_result = conn.execute((text(txt))).fetchall()
    # print("orm_data: ")
    cmp(orm_result, sql_result)


def test_two():
    sub_query = select(distinct(func.count(Boat.bid))).where(Boat.color == "red")
    data_sel = select(Sailor.sname, Sailor.sid)
    table_join = data_sel.join_from(Boat, Reservation)
    table_join = table_join.join(Sailor)
    stmt = table_join.where(Boat.color == "red")
    stmt = stmt.group_by(Sailor.sid).having(func.count(Boat.bid) == sub_query)
    orm_result = conn.execute(stmt).fetchall()
    txt = """select sailors.sname, sailors.sid from reserves
    inner join boats
        on boats.bid = reserves.bid
    inner join sailors
        on sailors.sid = reserves.sid
    where boats.color = 'red'
    group by sailors.sid
    having COUNT(boats.bid) = (select distinct COUNT(boats.bid) from boats where boats.color = 'red');
"""
    sql_result = conn.execute((text(txt))).fetchall()
    print(stmt)
    cmp(orm_result, sql_result)


def test_three():
    sub_query = select(Reservation.sid).join(Boat).where(Boat.color != "red")
    data_sel = select(Sailor.sname, Sailor.sid)
    table_join = data_sel.join_from(Reservation, Boat).join(Sailor)
    stmt = table_join.where(Boat.color == "red").where(Sailor.sid.not_in(sub_query))
    stmt = stmt.group_by(Sailor.sid)
    txt = """     select sailors.sname, sailors.sid from reserves r
        inner join boats b
            on b.bid=r.bid
```

```
66          inner join sailors
67              on sailors.sid = r.sid
68          where b.color = 'red' and sailors.sid not in (select r.sid from reserv
    es r inner join boats b on b.bid=r.bid where b.color != 'red')
69          group by sailors.sid
70  """
71      orm_result = conn.execute(stmt).fetchall()
72      sql_result = conn.execute((text(txt))).fetchall()
73      cmp(orm_result, sql_result)
74
75
76  def test_four():
77      stmt = select(Reservation.bid, func.count(Reservation.bid)).group_by(
78          Reservation.bid
79      )
80      stmt = stmt.order_by(func.count(Reservation.bid).desc())
81      txt = """
82      select reserves.bid, COUNT(reserves.bid)
83      from reserves group by reserves.bid order
84      by COUNT(reserves.bid) desc limit 1
85      """
86      orm_result = conn.execute(stmt.limit(1)).fetchall()
87      sql_result = conn.execute(text(txt)).fetchall()
88
89
90  # if __name__ == "__main__":
91  #     test = TestClass()
92  #     test.test_one()
```

A few things the Business is missing:

No way to verify a reservation is valid (A boat is available
and a sailor is not scheduled at the same time)

By extension, there is no way for a boat to be
available/in repairs/retired. This can cause issues.

No way to invoice/track total boat usage for billing by sailor/client.
In addition, no way to track boat hours for maintenence & support.

A simple extension to solve all of these would be a "status"
tag in the boat table, and "duration" field in the reservations.

This duration field could then be summed for each sailor weekly, creating a cost.

If different boats have different capabilities and therefore different rates,
A rate field for each boat would make aggregating for invoices very very easy.