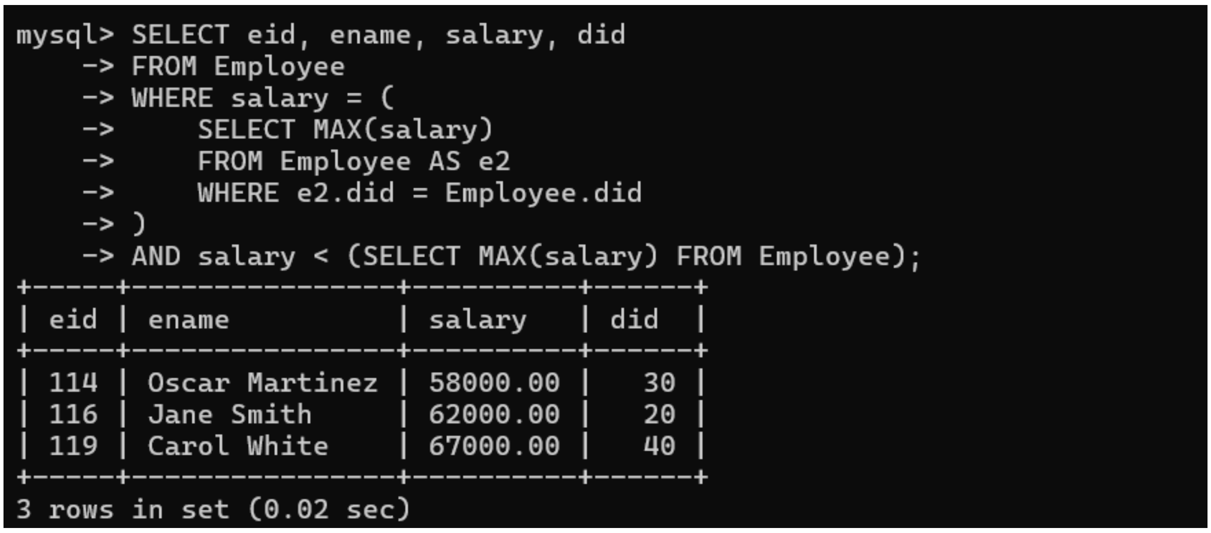
Objective:
The goal of this week is to practice queries on Aggregate functions like count, max, min, avg, sum and practice queries like nested queries/co- related queries using ANY, ALL, IN, Exists, NOT EXISTS, UNION, INTERSECT, group by and having etc.

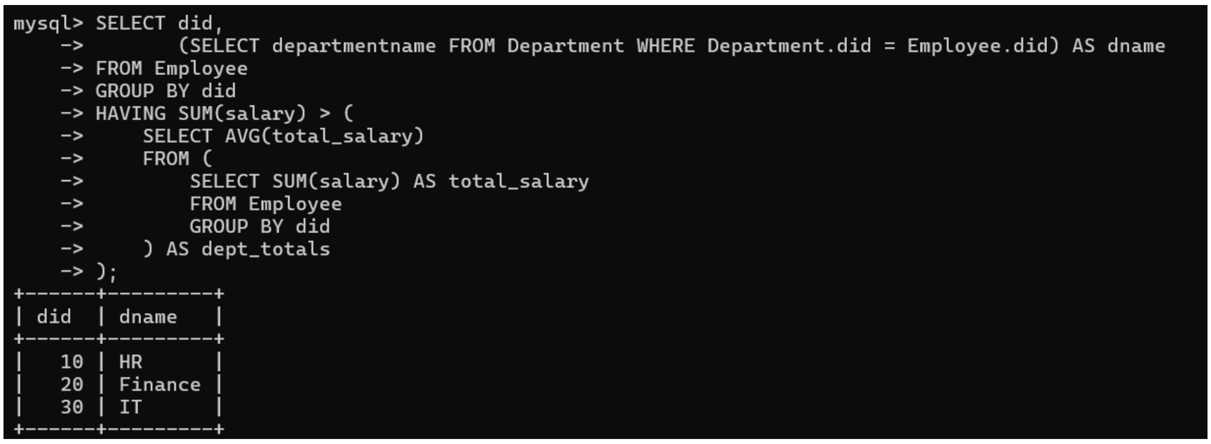
Task 1: Consider the following database tables and write the solution for the given queries.
Tables: Employee(eid, ename, salary, doj, comm,did) Department(did, departmentname, location)
Sample Data in Employees Table:

Q1). Find the employee who earns the maximum salary in their respective department but is not the highest-paid employee in the entire company.



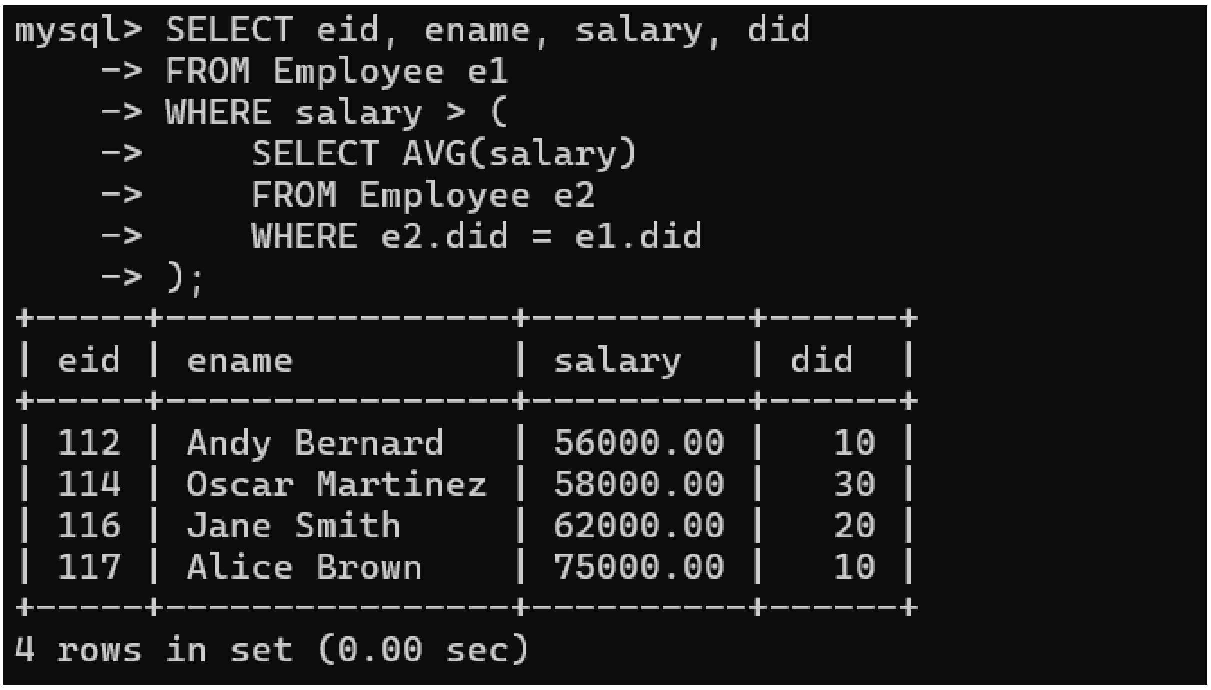
```
mysql> SELECT eid, ename, salary, did
    -> FROM Employee
    -> WHERE salary = (
    ->     SELECT MAX(salary)
    ->     FROM Employee AS e2
    ->     WHERE e2.did = Employee.did
    -> )
    -> AND salary < (SELECT MAX(salary) FROM Employee);
+-----+----------------+----------+------+
| eid | ename          | salary   | did  |
+-----+----------------+----------+------+
| 114 | Oscar Martinez | 58000.00 |   30 |
| 116 | Jane Smith     | 62000.00 |   20 |
| 119 | Carol White    | 67000.00 |   40 |
+-----+----------------+----------+------+
3 rows in set (0.02 sec)
```

Q2). Find departments where the total salary expenditure is greater than the average salary expenditure of all departments combined.



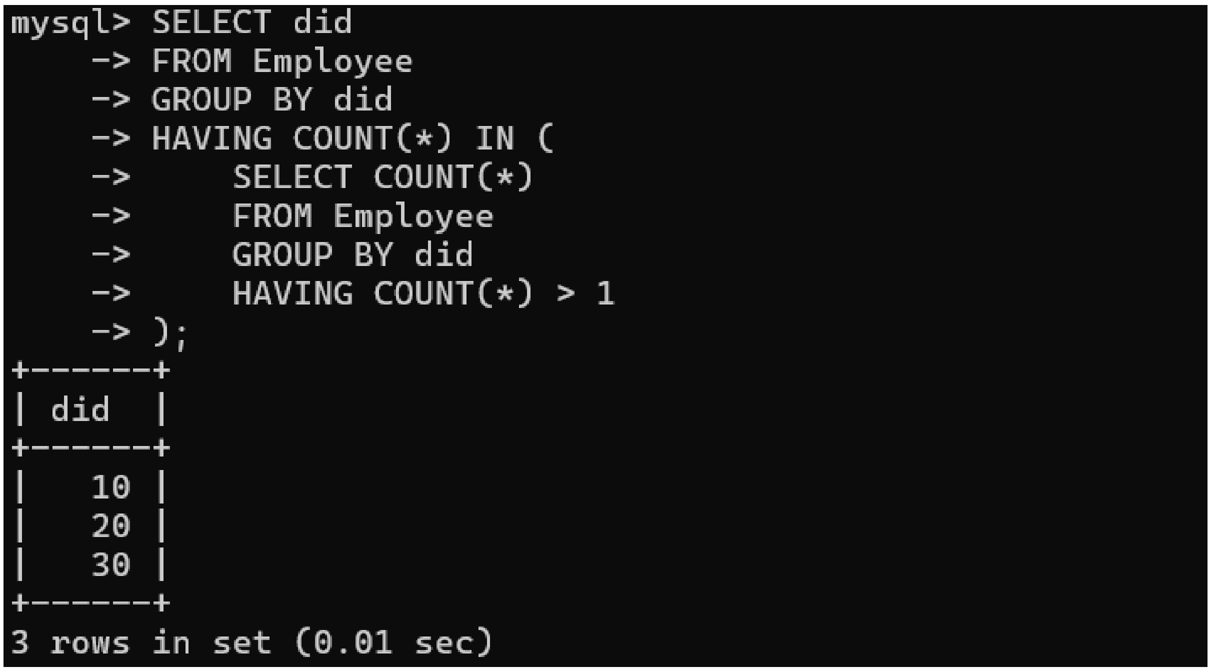
```
mysql> SELECT did,
    ->        (SELECT departmentname FROM Department WHERE Department.did = Employee.did) AS dname
    -> FROM Employee
    -> GROUP BY did
    -> HAVING SUM(salary) > (
    ->     SELECT AVG(total_salary)
    ->     FROM (
    ->         SELECT SUM(salary) AS total_salary
    ->         FROM Employee
    ->         GROUP BY did
    ->     ) AS dept_totals
    -> );
+------+---------+
| did  | dname   |
+------+---------+
|   10 | HR      |
|   20 | Finance |
|   30 | IT      |
+------+---------+
```

Q3) Find employees who earn more than the average salary of employees in the same department.



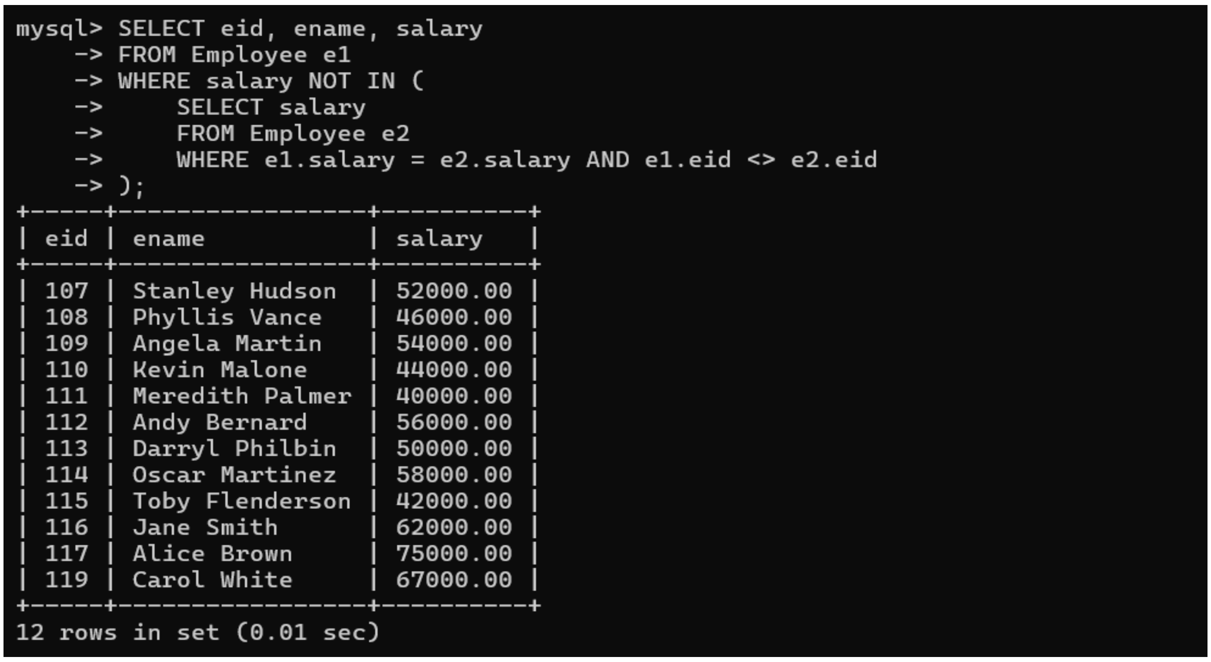
```
mysql> SELECT eid, ename, salary, did
    -> FROM Employee e1
    -> WHERE salary > (
    ->     SELECT AVG(salary)
    ->     FROM Employee e2
    ->     WHERE e2.did = e1.did
    -> );
+-----+----------------+----------+------+
| eid | ename          | salary   | did  |
+-----+----------------+----------+------+
| 112 | Andy Bernard   | 56000.00 |   10 |
| 114 | Oscar Martinez | 58000.00 |   30 |
| 116 | Jane Smith     | 62000.00 |   20 |
| 117 | Alice Brown    | 75000.00 |   10 |
+-----+----------------+----------+------+
4 rows in set (0.00 sec)
```

Q4). List all departments that have exactly the same number of employees as another department.



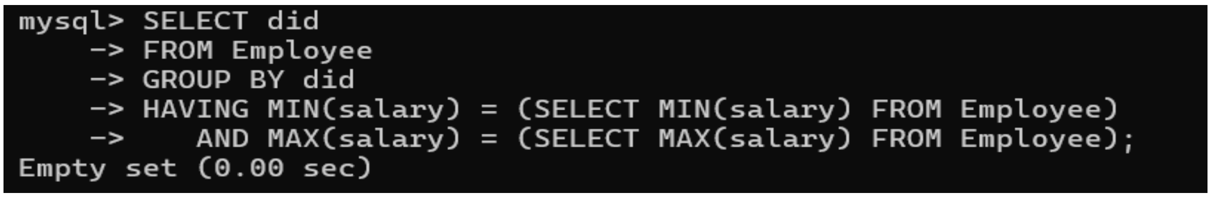
```
mysql> SELECT did
    -> FROM Employee
    -> GROUP BY did
    -> HAVING COUNT(*) IN (
    ->     SELECT COUNT(*)
    ->     FROM Employee
    ->     GROUP BY did
    ->     HAVING COUNT(*) > 1
    -> );
+------+
| did  |
+------+
|   10 |
|   20 |
|   30 |
+------+
3 rows in set (0.01 sec)
```

Q5). Find employees who do not share their salary with any other employee.



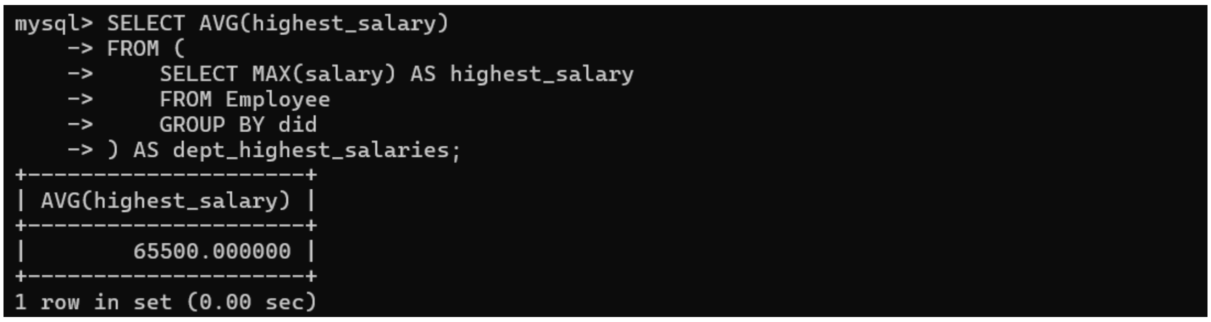
```
mysql> SELECT eid, ename, salary
    -> FROM Employee e1
    -> WHERE salary NOT IN (
    ->     SELECT salary
    ->     FROM Employee e2
    ->     WHERE e1.salary = e2.salary AND e1.eid <> e2.eid
    -> );
+-----+-----------------+----------+
| eid | ename           | salary   |
+-----+-----------------+----------+
| 107 | Stanley Hudson  | 52000.00 |
| 108 | Phyllis Vance   | 46000.00 |
| 109 | Angela Martin   | 54000.00 |
| 110 | Kevin Malone    | 44000.00 |
| 111 | Meredith Palmer | 40000.00 |
| 112 | Andy Bernard    | 56000.00 |
| 113 | Darryl Philbin  | 50000.00 |
| 114 | Oscar Martinez  | 58000.00 |
| 115 | Toby Flenderson | 42000.00 |
| 116 | Jane Smith      | 62000.00 |
| 117 | Alice Brown     | 75000.00 |
| 119 | Carol White     | 67000.00 |
+-----+-----------------+----------+
12 rows in set (0.01 sec)
```

Q6). Find the departments that have employees earning both the minimum and maximum salary in the company.



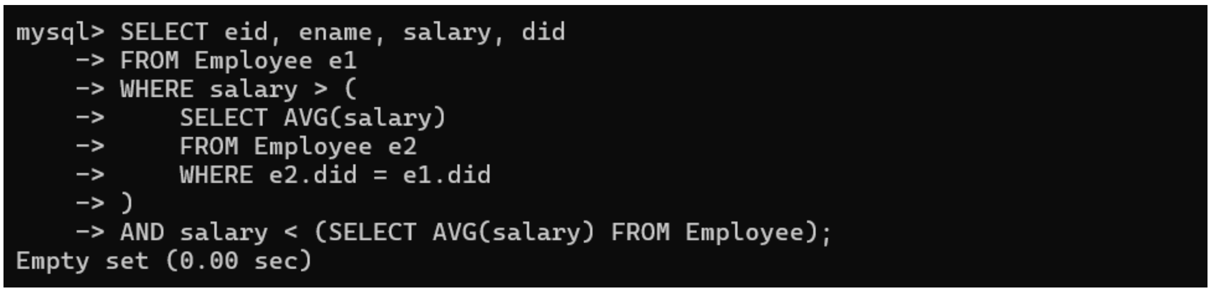
```
mysql> SELECT did
    -> FROM Employee
    -> GROUP BY did
    -> HAVING MIN(salary) = (SELECT MIN(salary) FROM Employee)
    ->    AND MAX(salary) = (SELECT MAX(salary) FROM Employee);
Empty set (0.00 sec)
```

Q7). Find the employee(s) with the highest salary in each department, and then find the average salary of these top earners across all departments.



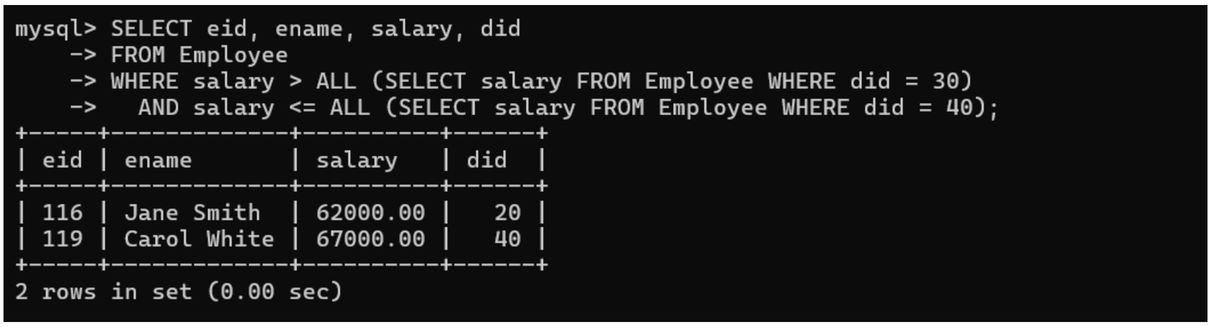
```
mysql> SELECT AVG(highest_salary)
    -> FROM (
    ->     SELECT MAX(salary) AS highest_salary
    ->     FROM Employee
    ->     GROUP BY did
    -> ) AS dept_highest_salaries;
+---------------------+
| AVG(highest_salary) |
+---------------------+
|        65500.000000 |
+---------------------+
1 row in set (0.00 sec)
```

Q8). List the employees who earn more than the average salary of their department, but less than the overall company average salary.



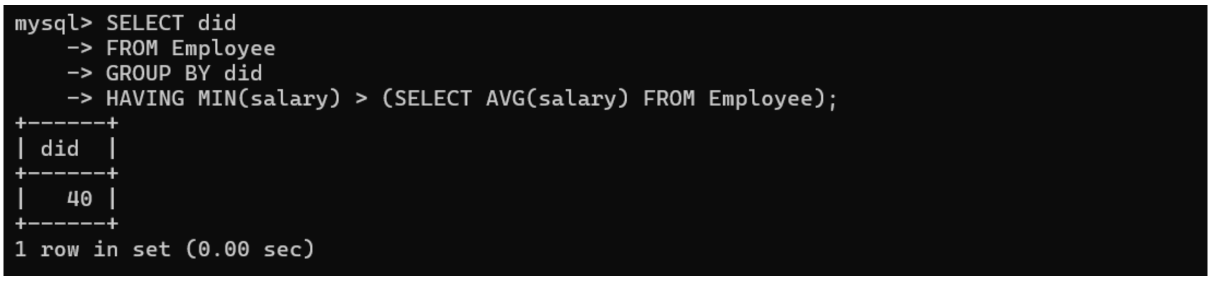
```
mysql> SELECT eid, ename, salary, did
    -> FROM Employee e1
    -> WHERE salary > (
    ->     SELECT AVG(salary)
    ->     FROM Employee e2
    ->     WHERE e2.did = e1.did
    -> )
    -> AND salary < (SELECT AVG(salary) FROM Employee);
Empty set (0.00 sec)
```

Q9). Find all employees whose salary is greater than the salary of every employee in department 30, but not greater than the salary of any employee in department 40.



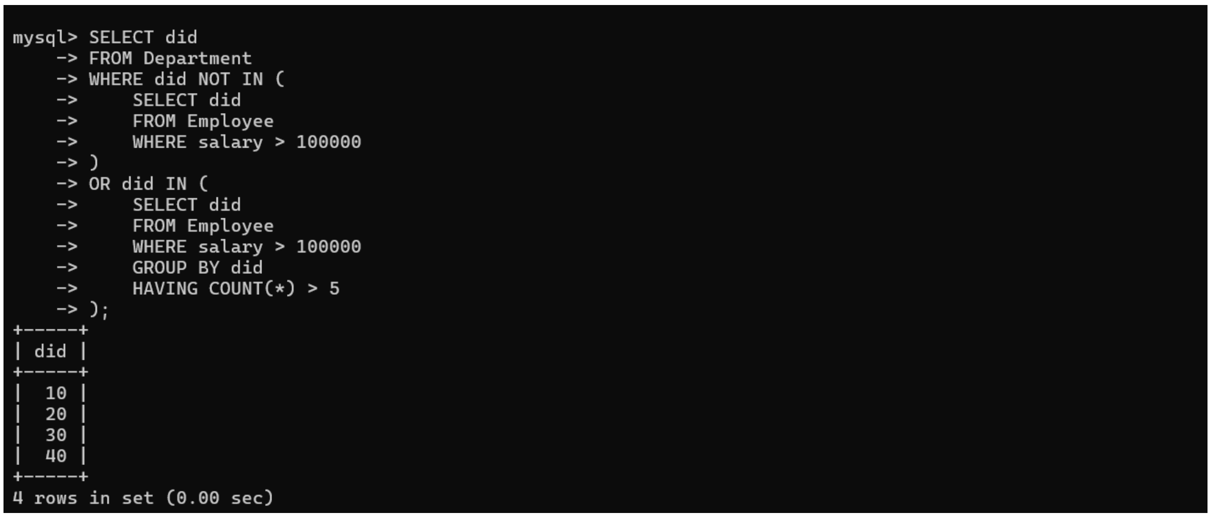
```
mysql> SELECT eid, ename, salary, did
    -> FROM Employee
    -> WHERE salary > ALL (SELECT salary FROM Employee WHERE did = 30)
    ->   AND salary <= ALL (SELECT salary FROM Employee WHERE did = 40);
+-----+-------------+----------+------+
| eid | ename       | salary   | did  |
+-----+-------------+----------+------+
| 116 | Jane Smith  | 62000.00 |   20 |
| 119 | Carol White | 67000.00 |   40 |
+-----+-------------+----------+------+
2 rows in set (0.00 sec)
```

Q10). List the departments where every employee has a salary greater than the overall average salary of all employees.



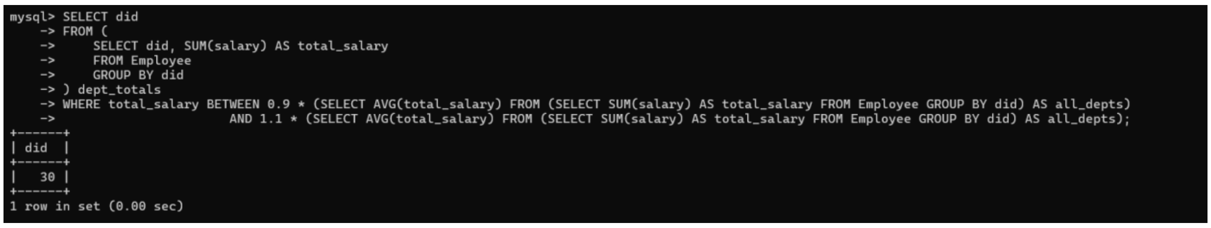
```
mysql> SELECT did
    -> FROM Employee
    -> GROUP BY did
    -> HAVING MIN(salary) > (SELECT AVG(salary) FROM Employee);
+------+
| did  |
+------+
|   40 |
+------+
1 row in set (0.00 sec)
```

Q11). Find employees who belong to departments that either have no employees with a salary above 100,000 or have more than 5 employees with a salary above 100,000.



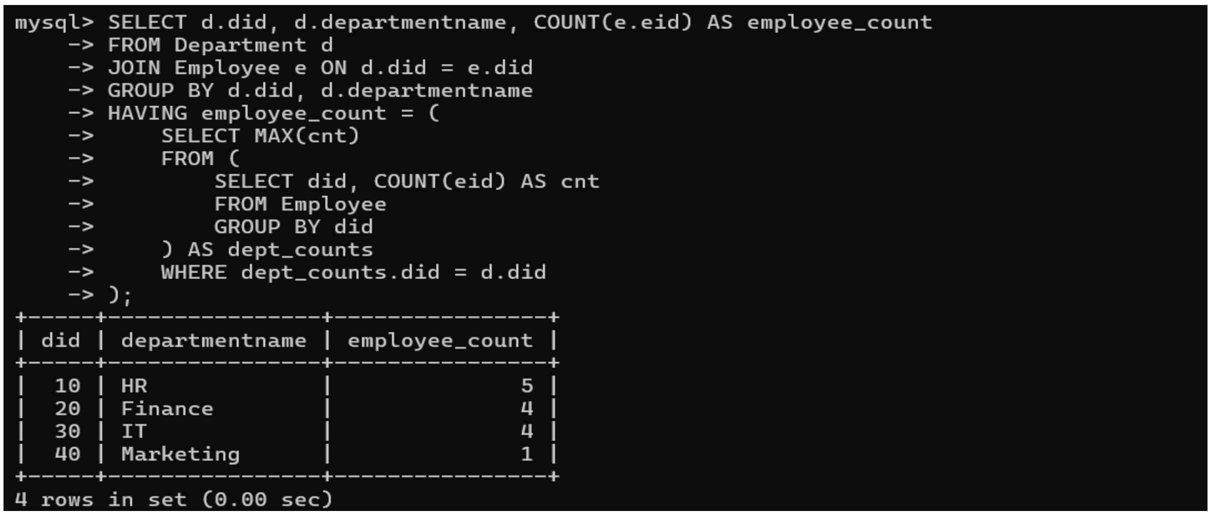
```
mysql> SELECT did
    -> FROM Department
    -> WHERE did NOT IN (
    ->     SELECT did
    ->     FROM Employee
    ->     WHERE salary > 100000
    -> )
    -> OR did IN (
    ->     SELECT did
    ->     FROM Employee
    ->     WHERE salary > 100000
    ->     GROUP BY did
    ->     HAVING COUNT(*) > 5
    -> );
+-----+
| did |
+-----+
|  10 |
|  20 |
|  30 |
|  40 |
+-----+
4 rows in set (0.00 sec)
```

Q12). Find all departments where the total salary expenditure is within 10% of the average total salary expenditure of all departments.



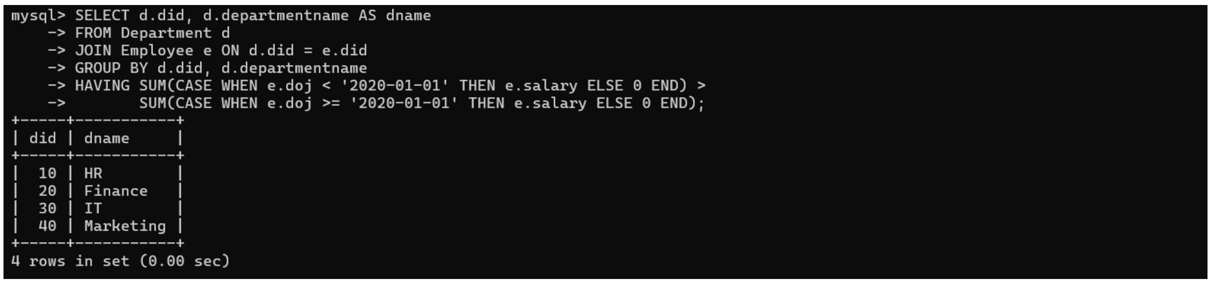
```
mysql> SELECT did
    -> FROM (
    ->     SELECT did, SUM(salary) AS total_salary
    ->     FROM Employee
    ->     GROUP BY did
    -> ) dept_totals
    -> WHERE total_salary BETWEEN 0.9 * (SELECT AVG(total_salary) FROM (SELECT SUM(salary) AS total_salary FROM Employee GROUP BY did) AS all_depts)
    ->                       AND 1.1 * (SELECT AVG(total_salary) FROM (SELECT SUM(salary) AS total_salary FROM Employee GROUP BY did) AS all_depts);
+------+
| did  |
+------+
|   30 |
+------+
1 row in set (0.00 sec)
```

Q13). For each department, find the most common job title and the number of employees with that title.



```
mysql> SELECT d.did, d.departmentname, COUNT(e.eid) AS employee_count
    -> FROM Department d
    -> JOIN Employee e ON d.did = e.did
    -> GROUP BY d.did, d.departmentname
    -> HAVING employee_count = (
    ->     SELECT MAX(cnt)
    ->     FROM (
    ->         SELECT did, COUNT(eid) AS cnt
    ->         FROM Employee
    ->         GROUP BY did
    ->     ) AS dept_counts
    ->     WHERE dept_counts.did = d.did
    -> );
+-----+----------------+----------------+
| did | departmentname | employee_count |
+-----+----------------+----------------+
|  10 | HR             |              5 |
|  20 | Finance        |              4 |
|  30 | IT             |              4 |
|  40 | Marketing      |              1 |
+-----+----------------+----------------+
4 rows in set (0.00 sec)
```

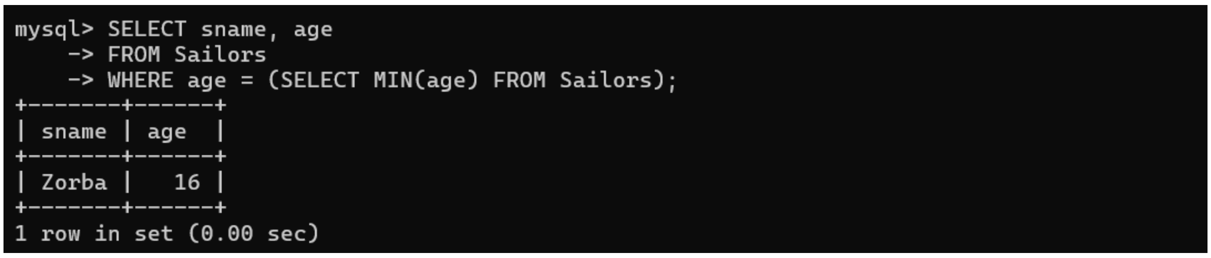
Q14). Identify departments where the sum of salaries for employees hired before 2020 is greater than the sum of salaries for employees hired after 2020.



```
mysql> SELECT d.did, d.departmentname AS dname
    -> FROM Department d
    -> JOIN Employee e ON d.did = e.did
    -> GROUP BY d.did, d.departmentname
    -> HAVING SUM(CASE WHEN e.doj < '2020-01-01' THEN e.salary ELSE 0 END) >
    ->        SUM(CASE WHEN e.doj >= '2020-01-01' THEN e.salary ELSE 0 END);
+-----+-----------+
| did | dname     |
+-----+-----------+
|  10 | HR        |
|  20 | Finance   |
|  30 | IT        |
|  40 | Marketing |
+-----+-----------+
4 rows in set (0.00 sec)
```

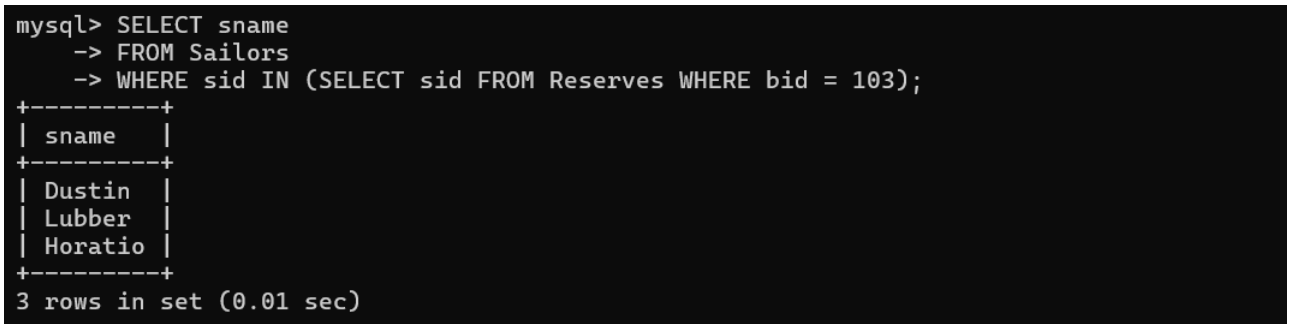
Task 2: Consider the following database tables and write the solution for the given queries.
Sailors (sid: integer, sname: string, rating: integer, age: real),
Boats (bid: integer, bname: string, color: string),
Reserves (sid: integer, bid: integer, day: date).

Q15) Find the name and the age of the youngest sailor.



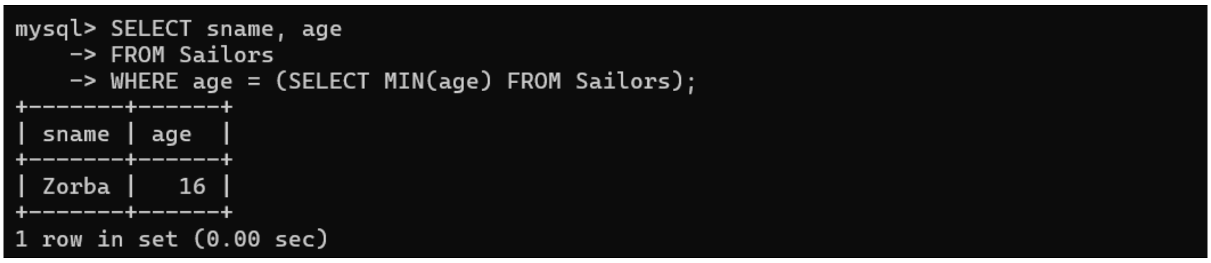
```
mysql> SELECT sname, age
    -> FROM Sailors
    -> WHERE age = (SELECT MIN(age) FROM Sailors);
+-------+------+
| sname | age  |
+-------+------+
| Zorba |   16 |
+-------+------+
1 row in set (0.00 sec)
```

Q16) Find the names of sailors who have reserved boat 103.



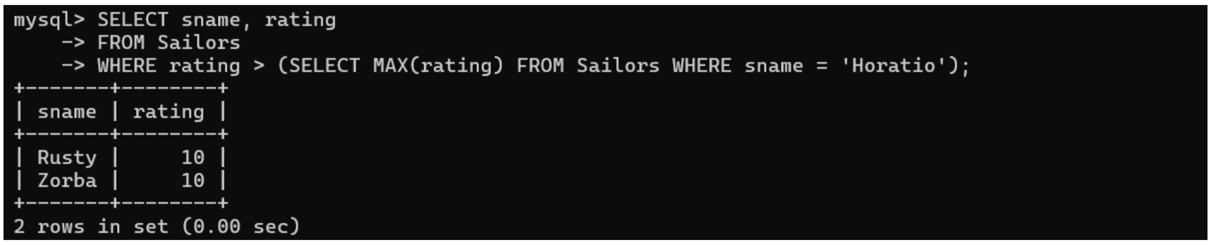
```
mysql> SELECT sname
    -> FROM Sailors
    -> WHERE sid IN (SELECT sid FROM Reserves WHERE bid = 103);
+---------+
| sname   |
+---------+
| Dustin  |
| Lubber  |
| Horatio |
+---------+
3 rows in set (0.01 sec)
```

Q17) Find the name and the age of the youngest sailor.



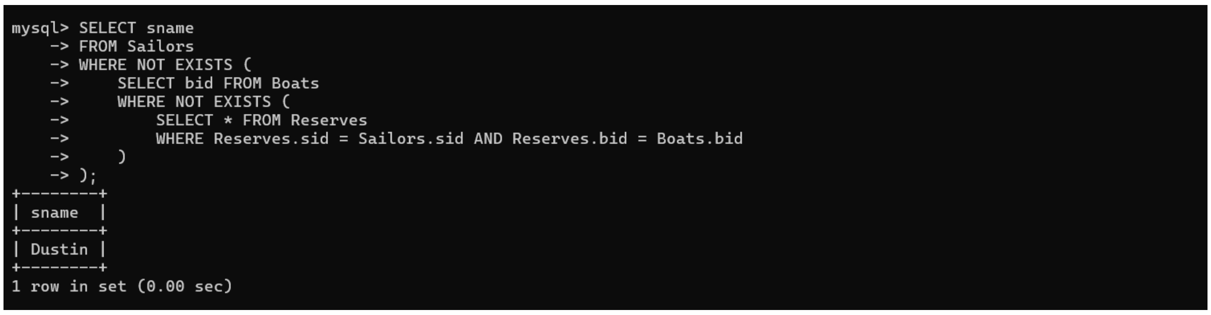
```
mysql> SELECT sname, age
    -> FROM Sailors
    -> WHERE age = (SELECT MIN(age) FROM Sailors);
+-------+------+
| sname | age  |
+-------+------+
| Zorba |   16 |
+-------+------+
1 row in set (0.00 sec)
```

Q18) Find the names and ratings of sailor whose rating is better than some sailor called Horatio



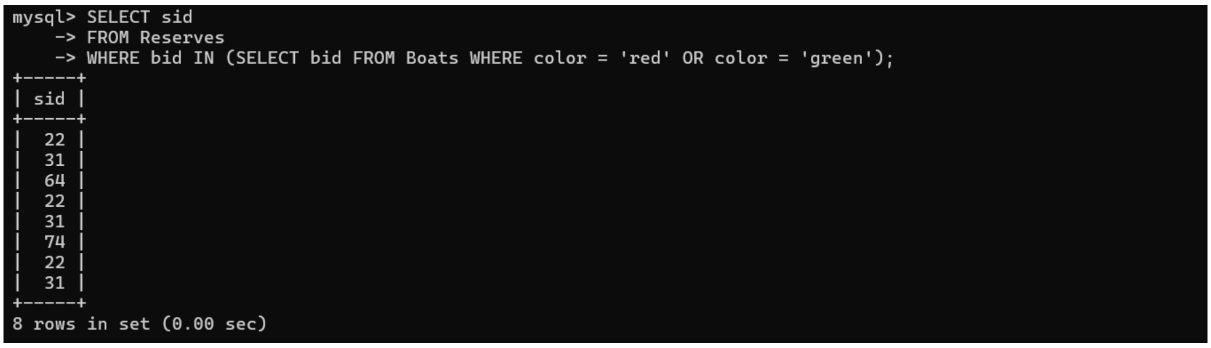
```
mysql> SELECT sname, rating
    -> FROM Sailors
    -> WHERE rating > (SELECT MAX(rating) FROM Sailors WHERE sname = 'Horatio');
+-------+--------+
| sname | rating |
+-------+--------+
| Rusty |     10 |
| Zorba |     10 |
+-------+--------+
2 rows in set (0.00 sec)
```

Q19) Find the names of sailors who have reserved all boats.



```
mysql> SELECT sname
    -> FROM Sailors
    -> WHERE NOT EXISTS (
    ->     SELECT bid FROM Boats
    ->     WHERE NOT EXISTS (
    ->         SELECT * FROM Reserves
    ->         WHERE Reserves.sid = Sailors.sid AND Reserves.bid = Boats.bid
    ->     )
    -> );
+--------+
| sname  |
+--------+
| Dustin |
+--------+
1 row in set (0.00 sec)
```

Q20) Find the ids of sailors who have reserved a red boat or a green boat.



```
mysql> SELECT sid
    -> FROM Reserves
    -> WHERE bid IN (SELECT bid FROM Boats WHERE color = 'red' OR color = 'green');
+-----+
| sid |
+-----+
|  22 |
|  31 |
|  64 |
|  22 |
|  31 |
|  74 |
|  22 |
|  31 |
+-----+
8 rows in set (0.00 sec)
```