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.

Task1: Aggregate Functions

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:

eid	ename	salary	doj	comm	did
106	Jim Halpert	48000	2017-09-18	400	10
107	Stanley Hudson	52000	2016-02-23	550	20
108	Phyllis Vance	46000	2015-08-11	350	30
109	Angela Martin	54000	2014-07-07	600	10
110	Kevin Malone	44000	2013-05-15	250	20
111	Meredith Palmer	40000	2012-03-28	150	30
112	Andy Bernard	56000	2011-01-10	700	10
113	Darryl Philbin	50000	2010-12-01	450	20
114	Oscar Martinez	58000	2009-11-17	800	30
115	Toby Flenderson	42000	2008-10-22	NULL	10
116	Jane Smith	62000	2012-03-28	400	20
117	Alice Brown	75000	2011-01-10	550	10
118	Bob Davis	48000	2010-12-01	350	30
119	Carol White	67000	2009-11-17	600	40

Sample data in Department Table:

DID	DepartmentName	Location
10	HR	New York
20	Finance	London

30 IT San Francisco

40 Marketing Chicago

Q1). Write a query to Count the number of employees in each department.

```
mysql> SELECT Department.departmentname, COUNT(Employee.eid) AS employee_count
    -> FROM Employee
    -> JOIN Department ON Employee.did = Department.did
    -> GROUP BY Department.departmentname;
+---------------+---------------+
| departmentname | employee_count |
+---------------+---------------+
| HR             |             5 |
| Finance        |             4 |
| IT             |             4 |
| Marketing      |             1 |
+---------------+---------------+
4 rows in set (0.02 sec)
```

Q2). Write a query to Find the maximum salary in the company.

```
mysql> SELECT MAX(salary) AS max_salary
    -> FROM Employee;
+------------+
| max_salary |
+------------+
|   75000.00 |
+------------+
1 row in set (0.01 sec)
```

Q3). Write a query to Find the minimum salary in each department.

```
mysql>
mysql> SELECT Department.departmentname, MIN(Employee.salary) AS min_salary
    -> FROM Employee
    -> JOIN Department ON Employee.did = Department.did
    -> GROUP BY Department.departmentname;
+---------------+------------+
| departmentname | min_salary |
+---------------+------------+
| HR             |   42000.00 |
| Finance        |   44000.00 |
| IT             |   40000.00 |
| Marketing      |   67000.00 |
+---------------+------------+
4 rows in set (0.00 sec)
```

Q4). Write a query to Calculate the average salary of employees.

```
mysql>
mysql> SELECT AVG(salary) AS average_salary
    -> FROM Employee;
+----------------+
| average_salary |
+----------------+
|   53000.000000 |
+----------------+
1 row in set (0.00 sec)
```

Q5). Write a query to Sum the total salaries of all employees.

```
mysql>
mysql> SELECT SUM(salary) AS total_salary
    -> FROM Employee;
+--------------+
| total_salary |
+--------------+
|    742000.00 |
+--------------+
1 row in set (0.00 sec)
```

Q6). Write a query to Count the number of employees in each department, but only for departments with more than 5 employees.

```
mysql> SELECT Department.departmentname, COUNT(Employee.eid) AS employee_count
    -> FROM Employee
    -> JOIN Department ON Employee.did = Department.did
    -> GROUP BY Department.departmentname
    -> HAVING COUNT(Employee.eid) > 5;
Empty set (0.00 sec)
```

Q7). Write a query to Find the average salary for each department, but only include job titles where the average salary is greater than 60,000.

```
mysql> SELECT Department.departmentname, AVG(Employee.salary) AS average_salary
    -> FROM Employee
    -> JOIN Department ON Employee.did = Department.did
    -> GROUP BY Department.departmentname
    -> HAVING AVG(Employee.salary) > 60000;
+---------------+----------------+
| departmentname | average_salary |
+---------------+----------------+
| Marketing      |   67000.000000 |
+---------------+----------------+
1 row in set (0.01 sec)
```

Q8). Write a query to Find the department with the second highest average salary.

```
mysql> SELECT Department.departmentname, AVG(Employee.salary) AS average_salary
    -> FROM Employee
    -> JOIN Department ON Employee.did = Department.did
    -> GROUP BY Department.departmentname
    -> ORDER BY average_salary DESC
    -> LIMIT 1 OFFSET 1;
+----------------+----------------+
| departmentname | average_salary |
+----------------+----------------+
| HR             |   55000.000000 |
+----------------+----------------+
1 row in set (0.01 sec)
```

Q9). Write a query to Find the employee who has the third highest salary in the company.

```
mysql>
mysql> SELECT eid, ename, salary
    -> FROM Employee
    -> WHERE salary = (
    ->     SELECT DISTINCT salary
    ->     FROM Employee
    ->     ORDER BY salary DESC
    ->     LIMIT 1 OFFSET 2
    -> );
+------+------------+----------+
| eid  | ename      | salary   |
+------+------------+----------+
|  116 | Jane Smith | 62000.00 |
+------+------------+----------+
1 row in set (0.01 sec)
```

Q10). Write a query to Union of employees from department 10 and department 20.

```
mysql> SELECT * FROM Employee WHERE did = 10
    -> UNION
    -> SELECT * FROM Employee WHERE did = 20;
+------+----------------+----------+------------+--------+------+
| eid  | ename          | salary   | doj        | comm   | did  |
+------+----------------+----------+------------+--------+------+
|  106 | Jim Halpert    | 48000.00 | 2017-09-18 | 400.00 |   10 |
|  109 | Angela Martin  | 54000.00 | 2014-07-07 | 600.00 |   10 |
|  112 | Andy Bernard   | 56000.00 | 2011-01-10 | 700.00 |   10 |
|  115 | Toby Flenderson| 42000.00 | 2008-10-22 |   NULL |   10 |
|  117 | Alice Brown    | 75000.00 | 2011-01-10 | 550.00 |   10 |
|  107 | Stanley Hudson | 52000.00 | 2016-02-23 | 550.00 |   20 |
|  110 | Kevin Malone   | 44000.00 | 2013-05-15 | 250.00 |   20 |
|  113 | Darryl Philbin | 50000.00 | 2010-12-01 | 450.00 |   20 |
|  116 | Jane Smith     | 62000.00 | 2012-03-28 | 400.00 |   20 |
+------+----------------+----------+------------+--------+------+
9 rows in set (0.01 sec)
```

Q11). Write a SQL query to identify employees who both work in department 10 and have a salary greater than 60,000.

```
mysql> SELECT eid, ename, salary
    -> FROM Employee
    -> WHERE did = 10 AND salary > 60000;
+------+-------------+----------+
| eid  | ename       | salary   |
+------+-------------+----------+
|  117 | Alice Brown | 75000.00 |
+------+-------------+----------+
1 row in set (0.00 sec)
```

Q12): Find the average salary of each department, but exclude departments where the minimum salary is less than 30,000.

```
mysql>
mysql> SELECT Department.departmentname, AVG(Employee.salary) AS average_salary
    -> FROM Employee
    -> JOIN Department ON Employee.did = Department.did
    -> GROUP BY Department.departmentname
    -> HAVING MIN(Employee.salary) >= 30000;
+----------------+----------------+
| departmentname | average_salary |
+----------------+----------------+
| HR             |   55000.000000 |
| Finance        |   52000.000000 |
| IT             |   48000.000000 |
| Marketing      |   67000.000000 |
+----------------+----------------+
4 rows in set (0.01 sec)
```

Q13): Find the total number of employees for job titles where the total salary paid is within 10% of the maximum salary paid for that job title.

```
mysql>
mysql> SELECT COUNT(eid) AS total_employees
    -> FROM Employee e
    -> WHERE salary >= (SELECT MAX(salary) * 0.9 FROM Employee WHERE ename = e.ename)
    ->   AND salary <= (SELECT MAX(salary) FROM Employee WHERE ename = e.ename);
+-----------------+
| total_employees |
+-----------------+
|              14 |
+-----------------+
1 row in set (0.01 sec)
```

Q14): Find the department where the difference between the highest and lowest salaries is the largest.

```
mysql> SELECT Department.departmentname, MAX(Employee.salary) - MIN(Employee.salary) AS salary_difference
    -> FROM Employee
    -> JOIN Department ON Employee.did = Department.did
    -> GROUP BY Department.departmentname
    -> ORDER BY salary_difference DESC
    -> LIMIT 1;
+----------------+-------------------+
| departmentname | salary_difference |
+----------------+-------------------+
| HR             |          33000.00 |
+----------------+-------------------+
1 row in set (0.01 sec)
```

Task2. Nested and Correlated Queries

Consider the following database tables and write the solution for the given queries Using ANY, ALL, IN, EXISTS, NOT EXISTS, UNION, INTERSECT.

Tables: Employee (eid, ename, salary, doj, comm, did) Department (did, dname, location)

Q15) Write a query to Find employees who earn more than the average salary:

```
mysql> SELECT eid, ename, salary
    -> FROM Employee
    -> WHERE salary > (SELECT AVG(salary) FROM Employee);
+------+----------------+----------+
| eid  | ename          | salary   |
+------+----------------+----------+
|  109 | Angela Martin  | 54000.00 |
|  112 | Andy Bernard   | 56000.00 |
|  114 | Oscar Martinez | 58000.00 |
|  116 | Jane Smith     | 62000.00 |
|  117 | Alice Brown    | 75000.00 |
|  119 | Carol White    | 67000.00 |
+------+----------------+----------+
6 rows in set (0.00 sec)
```

Q16). Write a query to Find departments that have more employees than the average department.

```
mysql> SELECT Department.departmentname
    -> FROM Department
    -> JOIN Employee ON Department.did = Employee.did
    -> GROUP BY Department.departmentname
    -> HAVING COUNT(Employee.eid) > (SELECT AVG(employee_count)
    ->                               FROM (SELECT COUNT(eid) AS employee_count
    ->                                     FROM Employee
    ->                                     GROUP BY did) AS dept_counts);
+----------------+
| departmentname |
+----------------+
| HR             |
| Finance        |
| IT             |
+----------------+
3 rows in set (0.01 sec)
```

Q17). Write a query to Find employees whose salary is greater than the salary of all employees in department 20

```
mysql> SELECT eid, ename, salary
    -> FROM Employee
    -> WHERE salary > ALL (SELECT salary FROM Employee WHERE did = 20);
+------+-------------+----------+
| eid  | ename       | salary   |
+------+-------------+----------+
|  117 | Alice Brown | 75000.00 |
|  119 | Carol White | 67000.00 |
+------+-------------+----------+
2 rows in set (0.01 sec)
```

Q18). Write a query to Find employees who earn more than anyone in department 10.

```
mysql> SELECT eid, ename, salary
    -> FROM Employee
    -> WHERE salary > ANY (SELECT salary FROM Employee WHERE did = 10);
+------+---------------+----------+
| eid  | ename         | salary   |
+------+---------------+----------+
|  106 | Jim Halpert   | 48000.00 |
|  107 | Stanley Hudson | 52000.00 |
|  108 | Phyllis Vance | 46000.00 |
|  109 | Angela Martin | 54000.00 |
|  110 | Kevin Malone  | 44000.00 |
|  112 | Andy Bernard  | 56000.00 |
|  113 | Darryl Philbin | 50000.00 |
|  114 | Oscar Martinez | 58000.00 |
|  116 | Jane Smith    | 62000.00 |
|  117 | Alice Brown   | 75000.00 |
|  118 | Bob Davis     | 48000.00 |
|  119 | Carol White   | 67000.00 |
+------+---------------+----------+
12 rows in set (0.00 sec)
```

Q19). Write a query to Find departments that have at least one employee with a salary greater than 50,000.

```
mysql> SELECT DISTINCT Department.departmentname
    -> FROM Department
    -> JOIN Employee ON Department.did = Employee.did
    -> WHERE Employee.salary > 50000;
+----------------+
| departmentname |
+----------------+
| Finance        |
| HR             |
| IT             |
| Marketing      |
+----------------+
4 rows in set (0.00 sec)
```

Q20). Write a query to Find departments that do not have any employees with a salary greater than 50,000.

```
mysql> SELECT Department.departmentname
    -> FROM Department
    -> WHERE Department.did NOT IN (
    ->     SELECT did
    ->     FROM Employee
    ->     WHERE salary > 50000
    -> );
Empty set (0.01 sec)
```

Task3: Nested and Correlated Queries for the University System

Consider the following database tables and write the solution for the given queries Using ANY, ALL, IN, EXISTS, NOT EXISTS, UNION, INTERSECT.

Sample Data for University System

Student (sid, sname, ccode, dob, address)

Course (ccode, cname, did, fees)

Department (did, dname, location)

Faculty (fid, fname, sal, designation, doj, did)

Nested Queries for the University System

Q21: Find students who have enrolled in courses offered by a specific department.

```
mysql> SELECT DISTINCT sname
    -> FROM Student
    -> WHERE ccode IN (
    ->     SELECT ccode
    ->     FROM Course
    ->     WHERE did = 1
    -> );
+---------+
| sname   |
+---------+
| Vinay   |
| Drushya |
+---------+
2 rows in set (0.01 sec)
```

Q22: Find the names of faculty members who teach courses that have a fee greater than the average course fee.

```
mysql> SELECT DISTINCT fname
    -> FROM Faculty
    -> WHERE did IN (
    ->     SELECT did
    ->     FROM Course
    ->     WHERE fees > (SELECT AVG(fees) FROM Course)
    -> );
Empty set (0.00 sec)
```

Q23: Find the names of students who have enrolled in all courses offered by a specific department.

```
mysql> SELECT sname
    -> FROM Student
    -> WHERE NOT EXISTS (
    ->     SELECT ccode
    ->     FROM Course
    ->     WHERE did = 1
    ->     EXCEPT
    ->     SELECT ccode
    ->     FROM Student
    ->     WHERE sid = Student.sid
    -> );
+-----------+
| sname     |
+-----------+
| Vinay     |
| Anirudh   |
| Drushya   |
| Sharath   |
| Vaishnavi |
+-----------+
5 rows in set (0.00 sec)
```

Q24: Find the names of departments where the average faculty salary is higher than the average student's course fee.

```
mysql> SELECT d.dname
    -> FROM Department d
    -> JOIN Faculty f ON d.did = f.did
    -> GROUP BY d.did, d.dname
    -> HAVING AVG(f.sal) > (SELECT AVG(c.fees) FROM Course c);
+-------+
| dname |
+-------+
| AIML  |
+-------+
1 row in set (0.00 sec)
```

Q25 Find the names of students who have enrolled in courses taught by faculty members with a salary greater than the average faculty salary.

```
mysql> SELECT s.sname
    -> FROM Student s
    -> JOIN Course c ON s.ccode = c.ccode
    -> JOIN Faculty f ON c.did = f.did
    -> WHERE f.sal > (SELECT AVG(sal) FROM Faculty)
    -> GROUP BY s.sid, s.sname;
+---------+
| sname   |
+---------+
| Sharath |
+---------+
1 row in set (0.00 sec)
```