Name – Adarsh kumar pandey
Exercise – 5

Reg. No . - 25MCMC29

employee (ID, emp_name, age, street, city)

works (ID, company_name, salary)

company (company_name, city)

2 . SQL Query Questions with Outputs

Q1. Use of union all operation findind employee who has age between 30 to 35 and 40 to 50

```
mysql> select emp_name FROM employee WHERE age BETWEEN 30 and 35
    -> UNION ALL
    -> select emp_name FROM employee WHERE age BETWEEN 40 and 50;
+------------+
| emp_name   |
+------------+
| Adarsh     |
| billu dada |
| Adarsh     |
| zaza       |
| ajit raja  |
+------------+
5 rows in set (0.00 sec)

mysql>
```

Q2. Use of intersection all to find all employee who are from ranchi and age > 30

```
mysql> select emp_name from employee WHERE city='ranchi'
    -> INTERSECT ALL
    -> select emp_name from employee WHERE age > 30 ;
+----------+
| emp_name |
+----------+
| Adarsh   |
| Adarsh   |
+----------+
2 rows in set (0.00 sec)

mysql>
```

Q3. Use of except all

```
mysql> select emp_name from employee WHERE city<>'ranchi'
    -> EXCEPT ALL
    -> select emp_name from employee WHERE age < 30;
+------------+
| emp_name   |
+------------+
| zaza       |
| ajit raja  |
| billu dada |
+------------+
3 rows in set (0.00 sec)

mysql>
```

Q4. show employee name where name is not null

```
mysql> SELECT emp_name from employee
    -> WHERE emp_name IS NOT NULL;
+------------+
| emp_name   |
+------------+
| zaza       |
| prince     |
| faran saju |
| Adarsh     |
| Akash Garv |
| akanksha   |
| ajit raja  |
| billu dada |
| Adarsh     |
+------------+
9 rows in set (0.00 sec)

mysql>
```

Q5. List details of eployee who's age is null or city is NULL

```
mysql> select * from employee
    -> WHERE age IS NULL or city IS NULL;
+---------+----------+------------+------+-----------+
| ID      | emp_name | street     | age  | city      |
+---------+----------+------------+------+-----------+
| 24soe33 | prince   | NULL       | NULL | manduadih |
| 25soe11 | NULL     | NULL       | NULL | NULL      |
| 25soe27 | akanksha | ram colony |   25 | NULL      |
+---------+----------+------------+------+-----------+
3 rows in set (0.00 sec)

mysql>
```

Q6. List all tuples who have null in any tupple

```
mysql> select * from employee
    -> WHERE emp_name is null or street is null or age is null or city is null;
+---------+----------+------------+------+-----------+
| ID      | emp_name | street     | age  | city      |
+---------+----------+------------+------+-----------+
| 23soe21 | NULL     | gomo       |   54 | chapra    |
| 24soe33 | prince   | NULL       | NULL | manduadih |
| 25soe11 | NULL     | NULL       | NULL | NULL      |
| 25soe27 | akanksha | ram colony |   25 | NULL      |
+---------+----------+------------+------+-----------+
4 rows in set (0.00 sec)

mysql>
```

Q7. Perform not operation

```
mysql> select distinct  emp_name from employee where emp_name IS NOT NULL;
+------------+
| emp_name   |
+------------+
| zaza       |
| prince     |
| faran saju |
| Adarsh     |
| Akash Garv |
| akanksha   |
| ajit raja  |
| billu dada |
+------------+
8 rows in set (0.01 sec)

mysql>
```

Q8. find average of salary

```
mysql> select avg(salary) as avg_sal from works;
+--------------+
| avg_sal      |
+--------------+
| 599333.3333 |
+--------------+
1 row in set (0.00 sec)

mysql>
```

Q9. find avg of salary without null

```
mysql> select avg(salary) as avg_sal from works where salary is not null;
+-------------+
| avg_sal     |
+-------------+
| 599333.3333 |
+-------------+
1 row in set (0.00 sec)
```

Q10. find name of employee who earns maximum salary

```
mysql> select e.emp_name , w.salary from employee e , works w where w.id = e.id and w.salary = (select max(salary) from works);
+----------+---------+
| emp_name | salary  |
+----------+---------+
| zaza     | 2000000 |
+----------+---------+
1 row in set (0.00 sec)

mysql>
```

Q11. Calculate total of salary

```
mysql> select sum(salary) as total_expenditure from works;
+--------------------+
| total_expenditure |
+--------------------+
|            3596000 |
+--------------------+
1 row in set (0.00 sec)

mysql>
```

Q12. Get name of employee who work in some organisation using exists

```
mysql> select e.emp_name from employee e where exists(select * from works w where w.id = e.id );
+------------+
| emp_name   |
+------------+
| zaza       |
| Akash Garv |
| faran saju |
| Adarsh     |
| ajit raja  |
| billu dada |
+------------+
6 rows in set (0.00 sec)
```

Q13. count no of tuples incluing null

```
mysql> select count(*) from employee;
+----------+
| count(*) |
+----------+
|       11 |
+----------+
1 row in set (0.00 sec)
```

Q14. count no of tuples excluding null

```
mysql> select count(emp_name) from employee where emp_name is not NULL;
+----------------+
| count(emp_name) |
+----------------+
|              9 |
+----------------+
1 row in set (0.01 sec)
```

Q15. use of min : aggrigate

```
mysql> SELECT MIN(salary) AS min_salary FROM works;
+------------+
| min_salary |
+------------+
|     240000 |
+------------+
1 row in set (0.00 sec)
```

Q16. use of as clause with aggrigate function

```
mysql> SELECT COUNT(emp_name) AS named_employees FROM employee;
+-----------------+
| named_employees |
+-----------------+
|               9 |
+-----------------+
1 row in set (0.00 sec)
```

Q17. use of where caluse with aggrigate function

```
mysql> SELECT AVG(salary) AS avg_salary FROM works WHERE company_name = 'flask';
+-------------+
| avg_salary  |
+-------------+
| 1150000.0000 |
+-------------+
1 row in set (0.00 sec)

mysql>
```

Q18. Use of COUNT(DISTINCT) with WHERE clause

```
mysql> SELECT COUNT(DISTINCT city) FROM employee WHERE age > 25;
+----------------------+
| COUNT(DISTINCT city) |
+----------------------+
|                    6 |
+----------------------+
1 row in set (0.00 sec)

mysql>
```

Q19. Use of COUNT with column and WHERE clause

```
mysql> SELECT COUNT(emp_name) FROM employee WHERE age > 25;
+-----------------+
| COUNT(emp_name) |
+-----------------+
|               6 |
+-----------------+
1 row in set (0.00 sec)

mysql>
```

Q20. Use of GROUP BY with AVG and AS alias

```
mysql> SELECT company_name, AVG(salary) AS avg_salary FROM works GROUP BY company_name;
+--------------+--------------+
| company_name | avg_salary   |
+--------------+--------------+
| flask        | 1150000.0000 |
| giio         |  420000.0000 |
| neo pvt      |  240000.0000 |
| roy grp      |  276000.0000 |
| sun          |  360000.0000 |
+--------------+--------------+
5 rows in set (0.00 sec)

mysql>
```

Q21.

```
mysql> SELECT company_name, AVG(salary) AS avg_salary FROM works GROUP BY company_name HAVING AVG(salary) > 300000;
+--------------+------------+
| company_name | avg_salary |
+--------------+------------+
| flask        | 916666.6667 |
| giio         | 400000.0000 |
| neo pvt      | 325000.0000 |
| sun          | 440000.0000 |
+--------------+------------+
4 rows in set (0.00 sec)

mysql>
```

Q22. Use of subquery in FROM clause with AS alias

```
mysql> SELECT emp_name, age FROM (SELECT emp_name, age FROM employee WHERE age IS NOT NULL) AS emp_details;
+--------------+------+
| emp_name     | age  |
+--------------+------+
| zaza         |   47 |
| NULL         |   54 |
| faran saju   |   26 |
| Adarsh       |   34 |
| Akash Garv   |   23 |
| akanksha     |   25 |
| ajit raja    |   45 |
| Rahul Sharma |   28 |
| Priya Singh  |   31 |
| Vikram Patel |   29 |
| Anjali Mehta |   35 |
| Sanjay Kumar |   42 |
| billu dada   |   30 |
| Adarsh       |   32 |
+--------------+------+
14 rows in set (0.00 sec)

mysql>
```

Q23. Use of subquery with MIN in WHERE clause

```
mysql> SELECT emp_name FROM employee WHERE age > (SELECT MIN(age) FROM employee);
+--------------+
| emp_name     |
+--------------+
| zaza         |
| NULL         |
| faran saju   |
| Adarsh       |
| akanksha     |
| ajit raja    |
| Rahul Sharma |
| Priya Singh  |
| Vikram Patel |
| Anjali Mehta |
| Sanjay Kumar |
| billu dada   |
| Adarsh       |
+--------------+
13 rows in set (0.00 sec)
```

Q24. Use of subquery with AVG in WHERE clause

```
mysql> SELECT emp_name FROM employee WHERE age > (SELECT AVG(age) FROM employee );
+--------------+
| emp_name     |
+--------------+
| zaza         |
| NULL         |
| ajit raja    |
| Anjali Mehta |
| Sanjay Kumar |
+--------------+
5 rows in set (0.00 sec)

mysql>
```

Q25. Use of scalar subquery in SELECT clause

```
mysql> SELECT (SELECT COUNT(*) FROM employee WHERE city = 'ranchi');
+-----------------------------------------------------+
| (SELECT COUNT(*) FROM employee WHERE city = 'ranchi') |
+-----------------------------------------------------+
|                                                   2 |
+-----------------------------------------------------+
1 row in set (0.00 sec)

mysql>
```

Q26. use of not in with nested subqueries

```
mysql> SELECT DISTINCT emp_name FROM employee
    -> WHERE city NOT IN (SELECT city FROM company
    -> WHERE company_name IN (SELECT company_name FROM works
    -> WHERE salary < 200000));
+--------------+
| emp_name     |
+--------------+
| zaza         |
| NULL         |
| prince       |
| faran saju   |
| Adarsh       |
| Akash Garv   |
| ajit raja    |
| billu dada   |
| Ravi Kumar   |
| Sneha Sharma |
| Mohit Das    |
| Anjali Mehta |
| Aryan Yadav  |
+--------------+
13 rows in set (0.00 sec)

mysql>
```

Q27. use of not in with literal values

```
mysql> -- 32. Find employees not in Delhi or Mumbai?
mysql> SELECT emp_name FROM employee
    -> WHERE city NOT IN ('delhi','mumbai');
+--------------+
| emp_name     |
+--------------+
| zaza         |
| NULL         |
| prince       |
| faran saju   |
| Adarsh       |
| ajit raja    |
| billu dada   |
| Adarsh       |
| Sneha Sharma |
| Anjali Mehta |
| Aryan Yadav  |
+--------------+
11 rows in set (0.00 sec)

mysql>
```

Q28. Use of row constructor with IN subquery

```
mysql> SELECT emp_name, age, city FROM employee
    -> WHERE (emp_name, age, city) IN (SELECT emp_name, age, city FROM employee
    -> WHERE age > 30);
+--------------+------+-----------+
| emp_name     | age  | city      |
+--------------+------+-----------+
| zaza         |   47 | hyderabad |
| Adarsh       |   34 | ranchi    |
| ajit raja    |   45 | gorakpur  |
| Adarsh       |   32 | ranchi    |
| Sneha Sharma |   41 | ranchi    |
| Mohit Das    |   38 | mumbai    |
| Aryan Yadav  |   31 | varanasi  |
+--------------+------+-----------+
7 rows in set (0.00 sec)

mysql>
```

Q29. Use of self-join with table aliases

```
mysql> SELECT DISTINCT A.emp_name, B.emp_name FROM employee AS A, employee AS B
    -> WHERE A.age > B.age AND A.emp_name LIKE 'A%';
+--------------+--------------+
| emp_name     | emp_name     |
+--------------+--------------+
| Aryan Yadav  | faran saju   |
| Anjali Mehta | faran saju   |
| Adarsh       | faran saju   |
| ajit raja    | faran saju   |
| ajit raja    | Adarsh       |
| Aryan Yadav  | Akash Garv   |
| Anjali Mehta | Akash Garv   |
| Adarsh       | Akash Garv   |
| ajit raja    | Akash Garv   |
| akanksha     | Akash Garv   |
| Aryan Yadav  | akanksha     |
| Anjali Mehta | akanksha     |
| Adarsh       | akanksha     |
| ajit raja    | akanksha     |
| Aryan Yadav  | billu dada   |
| Adarsh       | billu dada   |
| ajit raja    | billu dada   |
| Adarsh       | Adarsh       |
| Aryan Yadav  | Ravi Kumar   |
| Adarsh       | Ravi Kumar   |
| ajit raja    | Ravi Kumar   |
| ajit raja    | Sneha Sharma |
| ajit raja    | Mohit Das    |
| Aryan Yadav  | Anjali Mehta |
| Adarsh       | Anjali Mehta |
| ajit raja    | Anjali Mehta |
| Adarsh       | Aryan Yadav  |
| ajit raja    | Aryan Yadav  |
+--------------+--------------+
28 rows in set (0.01 sec)

mysql>
```

Q30. Use of self-join returning single column

```
mysql> SELECT DISTINCT A.emp_name FROM employee AS A, employee AS B
    -> WHERE A.age > B.age AND A.emp_name LIKE 'A%';
+--------------+
| emp_name     |
+--------------+
| Aryan Yadav  |
| Anjali Mehta |
| Adarsh       |
| ajit raja    |
| akanksha     |
+--------------+
5 rows in set (0.00 sec)

mysql>
```

Q31. Use of SOME comparison operator

```
mysql> SELECT emp_name FROM employee
    -> WHERE age > SOME (SELECT age FROM employee
    -> WHERE city = 'ranchi');
+--------------+
| emp_name     |
+--------------+
| zaza         |
| NULL         |
| Adarsh       |
| ajit raja    |
| Sneha Sharma |
| Mohit Das    |
+--------------+
6 rows in set (0.00 sec)

mysql>
```

Q32. Use of self-join with specific condition

```
mysql> SELECT DISTINCT A.emp_name FROM employee AS A, employee AS B
    -> WHERE A.age > B.age AND B.city = 'ranchi';
+--------------+
| emp_name     |
+--------------+
| zaza         |
| NULL         |
| Adarsh       |
| ajit raja    |
| Sneha Sharma |
| Mohit Das    |
+--------------+
6 rows in set (0.00 sec)

mysql>
```

Q33. Use of ALL comparison operator

```
mysql> SELECT emp_name FROM employee
    -> WHERE age > ALL (SELECT age FROM employee
    -> WHERE city = 'delhi');
+--------------+
| emp_name     |
+--------------+
| zaza         |
| NULL         |
| Adarsh       |
| ajit raja    |
| billu dada   |
| Adarsh       |
| Sneha Sharma |
| Mohit Das    |
| Aryan Yadav  |
+--------------+
9 rows in set (0.00 sec)

mysql>
```

Q34. Use of ALL with subquery


```
mysql> SELECT emp_name FROM employee
    -> WHERE age > ALL (SELECT age FROM employee
    -> WHERE city = 'delhi');
+--------------+
| emp_name     |
+--------------+
| zaza         |
| NULL         |
| Adarsh       |
| ajit raja    |
| billu dada   |
| Adarsh       |
| Sneha Sharma |
| Mohit Das    |
| Aryan Yadav  |
+--------------+
9 rows in set (0.00 sec)

mysql>
```


Q35. Use of ALL with subquery with where clause

```
mysql> SELECT emp_name FROM employee
    -> WHERE age > ALL (SELECT age FROM employee
    -> WHERE city = 'delhi');
+--------------+
| emp_name     |
+--------------+
| zaza         |
| NULL         |
| Adarsh       |
| ajit raja    |
| billu dada   |
| Adarsh       |
| Sneha Sharma |
| Mohit Das    |
| Aryan Yadav  |
+--------------+
9 rows in set (0.00 sec)

mysql>
```

Q36. Use of EXISTS with correlated subquery

```
mysql> SELECT emp_name FROM employee
    -> WHERE EXISTS (SELECT 1 FROM works
    -> WHERE employee.ID = works.ID AND salary > 300000);
+--------------+
| emp_name     |
+--------------+
| zaza         |
| faran saju   |
| billu dada   |
| Sneha Sharma |
| Mohit Das    |
+--------------+
5 rows in set (0.00 sec)

mysql>
```

Q37. Use of EXISTS with temp table

```
mysql> SELECT DISTINCT emp_name FROM employee e
    -> WHERE EXISTS (SELECT 1 FROM works w
    -> WHERE e.ID = w.ID AND w.salary > 300000);
+--------------+
| emp_name     |
+--------------+
| zaza         |
| faran saju   |
| billu dada   |
| Sneha Sharma |
| Mohit Das    |
+--------------+
5 rows in set (0.00 sec)

mysql>
```

Q38. Use of NOT EXISTS with condition

```
mysql> SELECT emp_name FROM employee e
    -> WHERE NOT EXISTS (SELECT 1 FROM works w
    -> WHERE e.ID = w.ID AND w.company_name = 'flask');
+--------------+
| emp_name     |
+--------------+
| NULL         |
| prince       |
| faran saju   |
| Adarsh       |
| NULL         |
| akanksha     |
| ajit raja    |
| billu dada   |
| Adarsh       |
| Sneha Sharma |
| Mohit Das    |
| Anjali Mehta |
| Aryan Yadav  |
+--------------+
13 rows in set (0.00 sec)

mysql>
```

Q39. Use of NOT EXISTS with comparison

```
mysql> SELECT DISTINCT emp_name FROM employee e
    -> WHERE NOT EXISTS (SELECT 1 FROM works w
    -> WHERE e.ID = w.ID AND w.salary > 400000);
+--------------+
| emp_name     |
+--------------+
| NULL         |
| prince       |
| Adarsh       |
| Akash Garv   |
| akanksha     |
| ajit raja    |
| billu dada   |
| Ravi Kumar   |
| Sneha Sharma |
| Anjali Mehta |
| Aryan Yadav  |
+--------------+
11 rows in set (0.00 sec)

mysql>
```

Q40. Use of subquery with COUNT(*) in WHERE

```
mysql> SELECT DISTINCT emp_name FROM employee e
    -> WHERE (SELECT COUNT(*) FROM works w
    -> WHERE e.ID = w.ID AND w.salary > 250000) >= 1;
+--------------+
| emp_name     |
+--------------+
| zaza         |
| faran saju   |
| Akash Garv   |
| ajit raja    |
| billu dada   |
| Ravi Kumar   |
| Sneha Sharma |
| Mohit Das    |
| Anjali Mehta |
+--------------+
9 rows in set (0.00 sec)

mysql>
```

Q41. Use of subquery with COUNT(company name) in WHERE

```
mysql> SELECT DISTINCT emp_name FROM employee e
    -> WHERE (SELECT COUNT(company_name) FROM works w
    -> WHERE e.ID = w.ID AND w.salary > 250000) >= 1;
+--------------+
| emp_name     |
+--------------+
| zaza         |
| faran saju   |
| Akash Garv   |
| ajit raja    |
| billu dada   |
| Ravi Kumar   |
| Sneha Sharma |
| Mohit Das    |
| Anjali Mehta |
+--------------+
9 rows in set (0.00 sec)

mysql>
```

Q42. use of with to get who pay above average total salary

```
mysql> WITH company_stats (company_name, total_salary) AS (SELECT company_name, SUM(salary)
    -> FROM works GROUP BY company_name),
    -> avg_stats (avg_salary) AS (SELECT AVG(total_salary) FROM company_stats)
    -> SELECT company_name FROM company_stats, avg_stats
    -> WHERE company_stats.total_salary > avg_stats.avg_salary;
+--------------+
| company_name |
+--------------+
| flask        |
+--------------+
1 row in set (0.00 sec)

mysql>
```

Q43. Use of correlated subquery in SELECT

```
mysql> SELECT e.emp_name, (SELECT COUNT(*) FROM works w
    -> WHERE w.ID = e.ID) AS work_count FROM employee e;
+-------------+------------+
| emp_name    | work_count |
+-------------+------------+
| zaza        |          1 |
| NULL        |          0 |
| prince      |          0 |
| faran saju  |          1 |
| Adarsh      |          1 |
| NULL        |          0 |
| Akash Garv  |          1 |
| akanksha    |          0 |
| ajit raja   |          1 |
| billu dada  |          1 |
| Adarsh      |          0 |
| Ravi Kumar  |          1 |
| Sneha Sharma |          1 |
| Mohit Das   |          1 |
| Anjali Mehta |          1 |
| Aryan Yadav |          1 |
+-------------+------------+
16 rows in set (0.01 sec)

mysql>
```

Q44. Use of multiple scalar subqueries in expression

```
mysql> SELECT (SELECT COUNT(*) FROM works) / (SELECT COUNT(*) FROM employee) AS ratio;
+--------+
| ratio  |
+--------+
| 0.6875 |
+--------+
1 row in set (0.00 sec)

mysql>
```

Q45. Use of scalar subqueries in arithmetic operation

```
mysql> SELECT (SELECT COUNT(*) FROM employee) - (SELECT COUNT(*) FROM works) AS difference;
+------------+
| difference |
+------------+
|          5 |
+------------+
1 row in set (0.00 sec)

mysql>
```

Q46. Use of DELETE with aggregate subquery

```
mysql> DELETE FROM works
    -> WHERE salary < (
    -> SELECT avg_sal
    -> FROM (
    -> SELECT AVG(salary) AS avg_sal
    -> FROM works
    -> ) AS temp
    -> );
Query OK, 9 rows affected (0.00 sec)

mysql>
```

Q47. Use of INSERT with VALUES clause

```
mysql> INSERT INTO employee
    -> VALUES ('25soe30', 'John Doe', 'Main Street', 28, 'Mumbai');
Query OK, 1 row affected (0.00 sec)

mysql>
```

Q48. Use of INSERT with column list

```
mysql> INSERT INTO employee (ID, emp_name, age)
    -> VALUES ('25soe31', 'Jane Smith', 32);
Query OK, 1 row affected (0.00 sec)

mysql>
```

Q49. Use of INSERT with SELECT subquery

```
mysql> INSERT INTO employee
    -> VALUES ('25soe32', NULL, 'Park Road', NULL, NULL);
Query OK, 1 row affected (0.00 sec)

mysql>
```

Q50. Use of UPDATE without WHERE clause

```
mysql> UPDATE works
    -> SET salary = salary * 1.1;
Query OK, 10 rows affected (0.00 sec)
Rows matched: 10  Changed: 10  Warnings: 0

mysql>
```

Q51. Use of UPDATE with WHERE condition

```
mysql> UPDATE works
    -> SET salary = salary * 1.1
    -> WHERE salary < 300000;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 0  Changed: 0  Warnings: 0

mysql>
```

Q52. Use of UPDATE with aggregate subquery

```
mysql> UPDATE works
    -> SET salary = salary * 1.1
    -> WHERE salary < (SELECT AVG(salary) FROM (SELECT salary FROM works) AS temp);
Query OK, 9 rows affected (0.00 sec)
Rows matched: 9  Changed: 9  Warnings: 0

mysql>
```

Q53. Use of UPDATE with CASE statement

```
mysql> UPDATE works
    -> SET salary = CASE WHEN salary <= 300000 THEN salary * 1.2
    -> ELSE salary * 1.1 END;
Query OK, 10 rows affected (0.00 sec)
Rows matched: 10  Changed: 10  Warnings: 0

mysql>
```

Now in the created employee database, provide the SQL queries for the following:

a. Find the ID and name of each employee who does not work for “University of Hyderabad”. (1 point)

```
mysql> select e.id , e.emp_name from employee e , works w
    -> where e.id = w.id and w.company_name <> 'university of hyderabad' and e.emp_name is not null;
+---------+-------------+
| id      | emp_name    |
+---------+-------------+
| 23soe1  | zaza        |
| 24soe33 | prince      |
| 24soe89 | faran saju  |
| 25soe2  | Akash Garv  |
| 25soe30 | John Doe    |
| 25soe31 | Jane Smith  |
| 25soe4  | ajit raja   |
| 25soe7  | billu dada  |
| 25soe9  | Adarsh      |
| 26soe1  | Ravi Kumar  |
| 26soe2  | Sneha Sharma |
| 26soe4  | Anjali Mehta |
| 26soe5  | Aryan Yadav |
+---------+-------------+
13 rows in set (0.00 sec)

mysql>
```

b. Find the ID and name of each employee who earns at least as much as every employee in the database. (1 point)

```
mysql> select e.id,e.emp_name  from employee e , works w
    -> where e.id = w.id and w.salary >= All(select salary from works);
+--------+----------+
| id     | emp_name |
+--------+----------+
| 23soe1 | zaza     |
+--------+----------+
1 row in set (0.01 sec)
```

c. Assume that companies may be located in several cities. Find the name of each company that is located in every city in which “SBI Bank” is located. (1 point)

```
Database changed
mysql> describe migrate_works;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| ID           | varchar(10) | NO   | PRI | NULL    |       |
| company_name | varchar(50) | YES  | MUL | NULL    |       |
| salary       | int         | YES  |     | NULL    |       |
| city         | varchar(150)| YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
4 rows in set (0.01 sec)

mysql> describe migrate_company;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| company_name | varchar(150)| NO   | PRI | NULL    |       |
| city         | varchar(50) | NO   | PRI | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql>
```

```
mysql> ^C
mysql> SELECT DISTINCT c1.company_name
    -> FROM migrate_works c1
    -> WHERE c1.company_name <> 'sbi'
    ->   AND c1.city IN (SELECT city FROM migrate_company WHERE company_name = 'sbi');
+--------------+
| company_name |
+--------------+
| giio         |
+--------------+
1 row in set (0.00 sec)

mysql>
```

d. Find the name of the company that has the most employees (or companies, in the case where there is a tie for the most). (1 point)

```
mysql> with company_count as (select company_name , count(*) as count from works group by company_name)
    -> select company_name , count from company_count where count = (select max(count) from company_count);
+--------------+-------+
| company_name | count |
+--------------+-------+
| flask        |     4 |
| giio         |     4 |
+--------------+-------+
2 rows in set (0.00 sec)

mysql>
```

e. Find the name of each company whose employees earn a higher salary, on average, than the average salary at “University of Hyderabad”. (1 point)

```
mysql> select e.emp_name from employee e , works w
    -> where w.id = e.id and w.salary > (select avg(salary) from works where company_name = 'university of hyderabad') and e.emp_name is not null;
+--------------+
| emp_name     |
+--------------+
| zaza         |
| faran saju   |
| John Doe     |
| Jane Smith   |
| billu dada   |
| Adarsh       |
| Ravi Kumar   |
| Sneha Sharma |
| Mohit Das    |
| Anjali Mehta |
| Aryan Yadav  |
+--------------+
11 rows in set (0.00 sec)

mysql>
```

4. Now add a new relation to the database.

manages(ID, manager_id)

```
mysql> create table manages (
    -> ID varchar(10) primary key ,
    -> manager_id varchar(10),
    -> foreign key (ID) references employee(ID)
    -> );
Query OK, 0 rows affected (0.02 sec)
```

6 . Now in the created employee database, provide the SQL queries for the following.
NB! These queries need a bit more exploration of SQL queries on the internet.

a. Think logically and add a new column to the employee table. Now update the rows in the table to also have some values for this new column. (1 point)

```
mysql> ALTER TABLE employee
    -> ADD COLUMN experience_years INT DEFAULT 0;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql>
```

```
mysql> UPDATE employee SET experience_years = 5 WHERE ID = '25soe2';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>
```

b. Change the attribute name ‘emp_name’ to ‘person_name’ in the employee table. (1 point)

```
mysql> ALTER TABLE employee
    -> CHANGE COLUMN emp_name person_name VARCHAR(100);
Query OK, 20 rows affected (0.03 sec)
Records: 20  Duplicates: 0  Warnings: 0

mysql>
```