RDBMS Assignment

1. **Create a database named employee, then import data_science_team.csv proj_table.csv and emp_record_table.csv into the employee database from the given resources.**

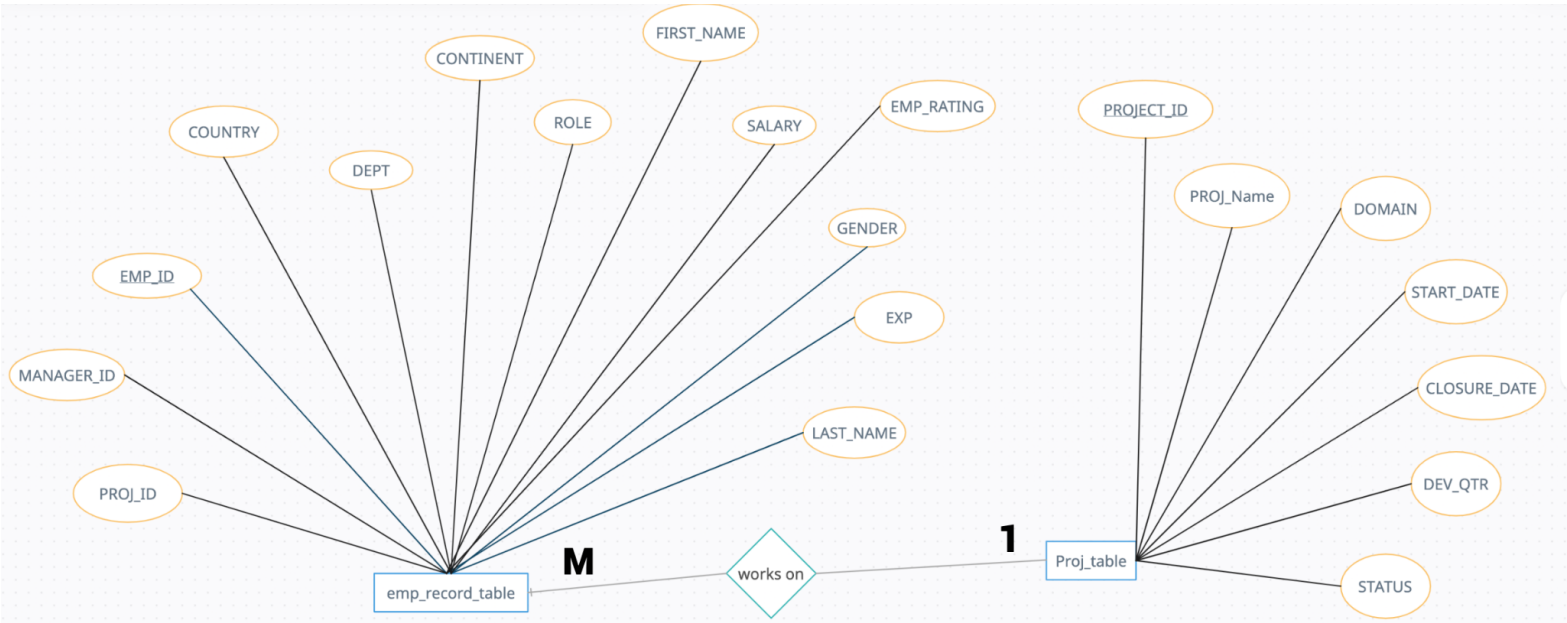
Query:

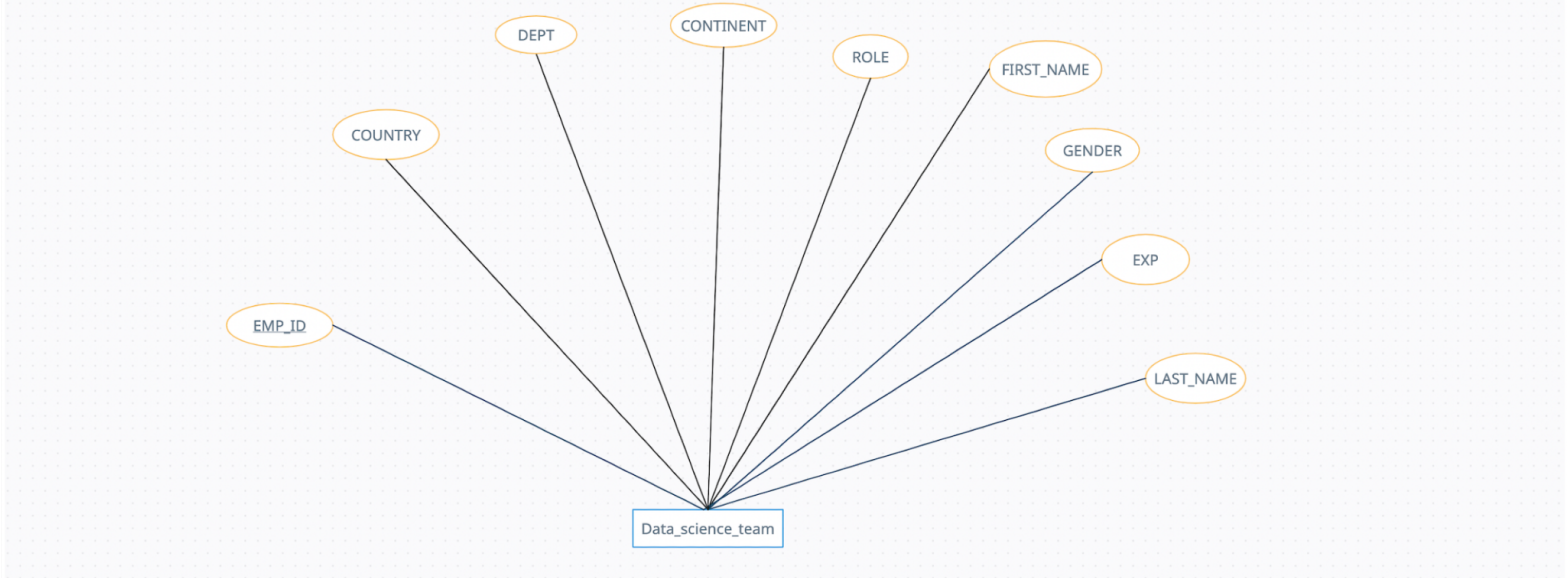
```
create database employee;

use employee;
```

- Subsequently I imported data_science_team.csv, proj_table.csv and emp_record_table.csv into the employee database from the given resources using MySQLWorkbench.

2. Create an ER diagram for the given employee database.





3. Write a query to fetch EMP_ID, FIRST_NAME, LAST_NAME, GENDER, and DEPARTMENT from the employee record table, and make a list of employees and details of their department.

Query:

```
SELECT emp_id, first_name, last_name, gender, dept

FROM emp_record_table;
```

4. Write a query to fetch EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPARTMENT, and EMP_RATING if the EMP_RATING is:

- **less than two**
- **greater than four**
- **between two and four**

Query:

4a:

```
SELECT emp_id, first_name, last_name, gender, dept, emp_rating

FROM emp_record_table

WHERE emp_rating<2;
```

4b:

```
SELECT emp_id, first_name, last_name, gender, dept, emp_rating

FROM emp_record_table
```

```
WHERE emp_rating>4;
```

4c:

```
SELECT emp_id, first_name, last_name, gender, dept, emp_rating
FROM emp_record_table
WHERE emp_rating BETWEEN 2 AND 4;
```

5. Write a query to concatenate the FIRST_NAME and the LAST_NAME of employees in the Finance department from the employee table and then give the resultant column alias as NAME.

Query:

```
SELECT CONCAT(first_name, ' ', last_name) as NAME
FROM emp_record_table
WHERE dept='Finance';
```

6. Write a query to list only those employees who have someone reporting to them. Also, show the number of reporters (including the President).

Query:

```
SELECT mgr.emp_id, mgr.first_name, mgr.last_name,
COUNT(e.emp_id)
```

```
FROM emp_record_table AS e INNER JOIN emp_record_table AS
mgr
ON e.manager_id = mgr.emp_id
GROUP BY mgr.emp_id, mgr.first_name, mgr.last_name;
```

7. Write a query to list down all the employees from the healthcare and finance departments using union. Take data from the employee record table.

Query:

```
SELECT emp_id, first_name, last_name, dept
FROM emp_record_table
WHERE dept='Healthcare'
UNION
SELECT emp_id, first_name, last_name, dept
FROM emp_record_table
WHERE dept='Finance';
```

8. Write a query to list down employee details such as EMP_ID, FIRST_NAME, LAST_NAME, ROLE, DEPARTMENT, and EMP_RATING grouped by dept. Also include the respective employee rating along with the max emp rating for the department.

Query:

```
SELECT emp_id, first_name, last_name, role, dept, emp_rating,
max(emp_rating) OVER(PARTITION BY dept) as max_rating_by_dept
FROM emp_record_table;
```

9. Write a query to calculate the minimum and the maximum salary of the employees in each role. Take data from the employee record table.

Query:

```
SELECT MIN(salary) as min_sal_by_role, MAX(salary) as
max_sal_by_role
FROM emp_record_table
GROUP BY role;
```

10. Write a query to assign ranks to each employee based on their experience. Take data from the employee record table.

Query:

```
SELECT DISTINCT emp_id, exp, DENSE_RANK() OVER(ORDER BY
exp DESC) AS rank_by_exp
FROM emp_record_table;
```

11. Write a query to create a view that displays employees in various countries whose salary is more than six thousand. Take data from the employee record table.

Query:

```
CREATE OR REPLACE VIEW sal_greater_six_K
AS SELECT emp_id, first_name, last_name, country, salary
FROM emp_record_table
WHERE salary>6000;
```

12. Write a nested query to find employees with experience of more than ten years. Take data from the employee record table.

Query:

```
SELECT *
FROM emp_record_table
WHERE emp_id IN
(SELECT emp_id
FROM emp_record_table
WHERE exp>10);
```

13. Write a query to create a stored procedure to retrieve the details of the employees whose experience is more than three years. Take data from the employee record table.

Query:

```
delimiter $$
create procedure emp_details()
begin
    select * from emp_record_table
    where exp>3;
end$$
delimiter ;
call emp_details();
```

14. Write a query using stored functions in the project table to check whether the job profile assigned to each employee in the data science team matches the organization's set standard.
The standard being:

For an employee with experience less than or equal to 2 years assign 'JUNIOR DATA SCIENTIST', For an employee with the experience of 2 to 5 years assign 'ASSOCIATE DATA SCIENTIST',

For an employee with the experience of 5 to 10 years assign 'SENIOR DATA SCIENTIST',

For an employee with the experience of 10 to 12 years assign 'LEAD DATA SCIENTIST',

For an employee with the experience of 12 to 16 years assign 'MANAGER'.

Query:

```
delimiter $$
create function emp_details() returns tinyint(1) deterministic
begin
    declare v_exp int default 0;
    declare v_role varchar(50) default "";
    declare finished int default 0;
    declare dummy_cursor cursor for
        select exp, role from emp_record_table;
    declare continue handler for not found
        set finished=1;
    open dummy_cursor;
    check_role: loop
        fetch dummy_cursor into v_exp, v_role;
```

```
            if finished = 1 then
                leave check_role;
            end if;
            if (v_exp<=2 and v_role!='JUNIOR DATA SCIENTIST') then
                return false;
            elseif (v_exp>2 and v_exp<=5 and v_role!='ASSOCIATE
DATA SCIENTIST') then
                return false;
            elseif (v_exp>5 and v_exp<=10 and v_role!='SENIOR DATA
SCIENTIST') then
                return false;
            elseif (v_exp>10 and v_exp<=12 and v_role!='LEAD DATA
SCIENTIST') then
                return false;
            elseif (v_exp>12 and v_exp<=16 and v_role!='MANAGER')
 then
                return false;
            end if;

    end loop check_role;
    close dummy_cursor;
```

```
    return true;
  end$$
  delimiter ;


  delimiter $$
  create procedure helper_procedure()
  begin
        if emp_details() then
              select 'The job profile assigned to each employee
  in the data science team matches the organization’s set standard.' as
message;
        else
              select 'The job profile assigned to each employee
  in the data science team does not match the organization’s set
standard.' as message;
        end if;
  end$$
  delimiter ;
  call helper_procedure();
```

15. Create an index to improve the cost and performance of the query to find the employee whose FIRST_NAME is ‘Eric’ in the employee table after checking the execution plan.

Query:

```
create index ename_index
on emp_record_table(first_name);
select *
from emp_record_table
where first_name = 'Eric';
```

16. Write a query to calculate the bonus for all the employees, based on their ratings and salaries (Use the formula: 5% of salary * employee rating).

Query:

```
select emp_id, emp_rating, salary, (0.05*salary*emp_rating) as bonus
from emp_record_table;
```

17. Write a query to calculate the average salary distribution based on the continent and country. Take data from the employee record table.

Query:

```
select distinct continent, avg(salary) over(partition by continent) as avg_sal_by_continent,
country, avg(salary) over(partition by country) as avg_sal_by_country
from emp_record_table;
```