ScienceQtech Employee Performance Mapping.

SQL Training Course-end Project 1

```
CREATE DATABASE employee;


CREATE TABLE emp_record (
                        emp_id VARCHAR(6) not null PRIMARY KEY,
                        f_name VARCHAR(10) not null,
                        l_name VARCHAR(10) not null,
                        gender VARCHAR(10) not null,
                        role VARCHAR(30) not null,
                        dept VARCHAR(15) not null,
                        exp INT not null,
        country VARCHAR(15) not null,
                        continent VARCHAR(15) not null,
                        salary INT not null,
                        emp_rating INT not null,
                        manager_id VARCHAR(5),
                        proj_id varchar(5));
use employee;
DESCRIBE emp_record;



CREATE TABLE proj_table (
                        proj_id VARCHAR(5) not null PRIMARY KEY,
                        proj_name VARCHAR(30) not null,
                        domain VARCHAR(15) not null,
                        start_date DATE not null,
                        closure_date DATE not null,
                        dev_qtr VARCHAR(4) not null,
                        status VARCHAR(10) not null);
```

```
describe proj_table;

CREATE TABLE data_sci_team (
                emp_id VARCHAR(6) not null PRIMARY KEY,
                f_name VARCHAR(10) not null,
                l_name VARCHAR(10) not null,
                gender VARCHAR(10) not null,
                role VARCHAR(30) not null,
                dept VARCHAR(15) not null,
                exp INT not null,
                country VARCHAR(15) not null,
                continent VARCHAR(15) not null);

describe data_sci_team;



SELECT emp_id, first_name, last_name, gender, dept
FROM employee.emp_record_table;

#EMP_RATING is Less than two
SELECT emp_id, first_name, last_name, gender, dept, emp_rating
FROM employee.emp_record_table
WHERE emp_rating < 2;

#EMP_RATING is Greater than four
SELECT emp_id, first_name, last_name, gender, dept, emp_rating
FROM employee.emp_record_table
WHERE emp_rating > 4;

#EMP_RATING is Between two and four
SELECT emp_id, first_name, last_name, gender, dept, emp_rating
```

```
FROM employee.emp_record_table

WHERE emp_rating BETWEEN 2 AND 4;
```

#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.

```
SELECT CONCAT(FIRST_NAME,' ',LAST_NAME) AS NAME

FROM emp_record_table

WHERE DEPT = "FINANCE";
```

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

```
SELECT m.EMP_ID,m.FIRST_NAME,m.LAST_NAME,m.ROLE,

m.EXP,COUNT(e.EMP_ID) as "EMP_COUNT"

FROM emp_record_table m

INNER JOIN emp_record_table e

ON m.EMP_ID = e.MANAGER_ID

GROUP BY m.EMP_ID

ORDER BY m.EMP_ID;
```

```
select role,manager_ID,count(*)

FROM employee.emp_record_table

GROUP BY emp_ID

ORDER BY emp_ID;
```

ISSUE ERROR above query

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

```
SELECT first_name, last_name, dept

FROM emp_record_table

WHERE dept =‘HEALTHCARE’

UNION

SELECT first_name, last_name, dept

FROM employee.emp_record_table

WHERE dept =‘FINANCE’

order by dept,emp_ID;
```

#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.

```
SELECT emp_id, first_name, last_name, role, dept, emp_rating, emp_rating AS max_rating

FROM employee.emp_record_table

WHERE (dept, emp_rating)

IN (SELECT dept, MAX(emp_rating) FROM employee.emp_record_table GROUP By dept)

ORDER BY dept ASC;
```

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

```
SELECT role, MIN(salary) AS minSalary, MAX(salary) AS maxSalary

FROM employee.emp_record_table

GROUP BY role;
```

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

```
SELECT first_name, last_name, exp as experience,
DENSE_RANK() OVER (ORDER BY exp DESC) exp_rank
FROM employee.emp_record_table;
```

#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.

#ERROR RRRRRRR

```
CREATE VIEW 6K_salary AS
SELECT emp_id, first_name, last_name, country, salary
FROM emp_record_table
WHERE salary > 6000;

SELECT * FROM 6k_salary;
```

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

```
SELECT emp_id, first_name, last_name, exp
FROM employee.emp_record_table
WHERE exp IN (
                    SELECT exp
                    FROM employee.emp_record_table
                    WHERE exp > 10
                    );
```

#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.

```
#ERRORRRRRRRRR
DELIMITER //
CREATE PROCEDURE Employee3()
BEGIN
	SELECT * FROM employee.emp_record_table
	WHERE exp > 3;
END //
DELIMITER ;
CALL Employee3;
```

#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.

```
#ERROR RRRRRRR
DELIMITER //
CREATE PROCEDURE check_role()
BEGIN
	SELECT * FROM employee.emp_record_table
	CASE
		WHEN exp <= 2 THEN SET role = 'JUNIOR DATA SCIENTIST';
		WHEN exp BETWEEN 3 AND 5 THEN SET role = 'ASSOCIATE DATA SCIENTIST';
		WHEN exp BETWEEN 6 AND 10 THEN SET role = 'SENIOR DATA SCIENTIST';
		WHEN exp BETWEEN 11 AND 12 THEN SET role = 'LEAD DATA SCIENTIST';
		WHEN exp BETWEEN 13 AND 16 THEN SET role = 'MANAGER';
		ELSE SET role = 'all good';
	END CASE;
END //
```

```
DELIMITER ;



#. 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.

#EROOR RRRRR

ALTER TABLE employee.emp_record_table ADD INDEX firstname_index (first_name);

SELECT * FROM employee.emp_record_table WHERE first_name = ‘Eric’;



#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).

SELECT first_name, last_name, salary, ((salary * .05)*emp_rating) AS bonus

FROM employee.emp_record_table;



#Write a query to calculate the average salary distribution based on the continent and country. Take data from the

#employee record table.

SELECT continent, AVG(salary)

FROM employee.emp_record_table

GROUP BY continent

ORDER BY continent ASC;



#END of Program
```