**Employee Performance Index**

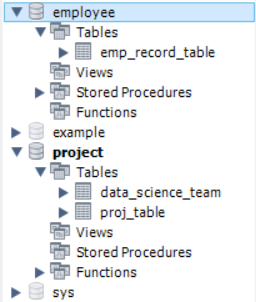
1. Create a database named project and employee, then import data\_science\_team.csv and proj\_table.csv into the project database and emp\_record\_table.csv into the employee database from the given resources.

**SQL code:**

CREATE DATABASE IF NOT EXISTS project;

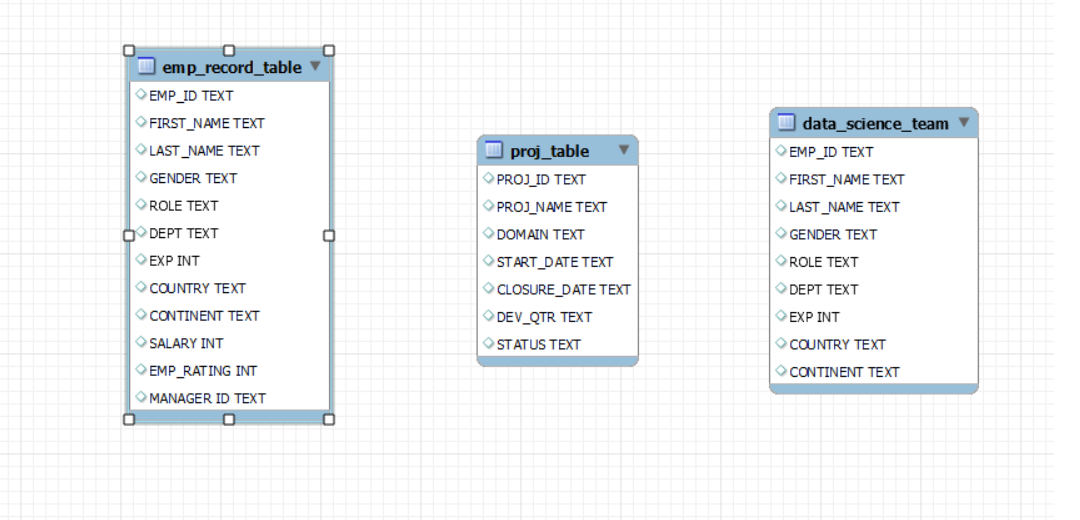
**Output:**

The following output shows the imported tables in the database.

****

1. Create an ER diagram for the given **project** and the **employee** databases.

**Output:**

****

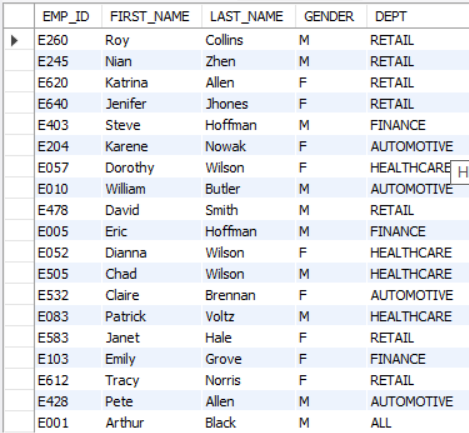
1. Write a query to fetch EMP\_ID, FIRST\_NAME, LAST\_NAME, GENDER, and DEPARTMENT from the employee record table, and make a list of employees and details of their department.

**SQL code:**

SELECT EMP\_ID, FIRST\_NAME, LAST\_NAME, GENDER, DEPT

FROM employee.emp\_record\_table;

**Output:**



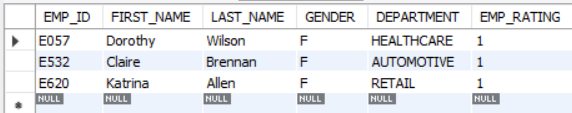
1. Write a query to fetch EMP\_ID, FIRST\_NAME, LAST\_NAME, GENDER, DEPARTMENT, and EMP\_RATING if the EMP\_RATING is:

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

**SQL code:**

SELECT EMP\_ID, FIRST\_NAME, LAST\_NAME, GENDER, DEPT, EMP\_RATING

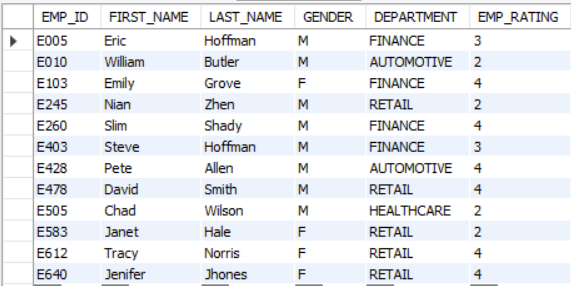
FROM employee.emp\_record\_table

WHERE EMP\_RATING < 2;

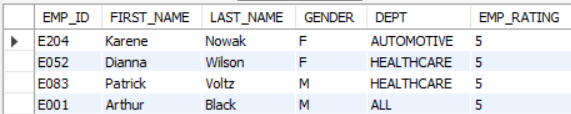
SELECT EMP\_ID, FIRST\_NAME, LAST\_NAME, GENDER, DEPT, EMP\_RATING

FROM employee.emp\_record\_table

WHERE EMP\_RATING between 2 and 4;



SELECT EMP\_ID, FIRST\_NAME, LAST\_NAME, GENDER, DEPT, EMP\_RATING  
FROM employee.emp\_record\_table  
WHERE EMP\_RATING > 4;



1. 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.

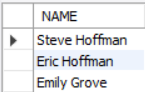
**SQL code:**

SELECT CONCAT\_WS(' ', FIRST\_NAME, LAST\_NAME) AS `NAME`

FROM employee.emp\_record\_table

WHERE DEPT = "FINANCE";

**Output:**

****

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

**SQL code:**

SELECT

m.EMP\_ID, m.FIRST\_NAME, m.LAST\_NAME, m.ROLE,

m.EXP, m.DEPT, COUNT(e.EMP\_ID) as "EMP\_COUNT"

FROM

employee.emp\_record\_table m

INNER JOIN employee.emp\_record\_table e

ON m.EMP\_ID = e.MANAGER\_ID

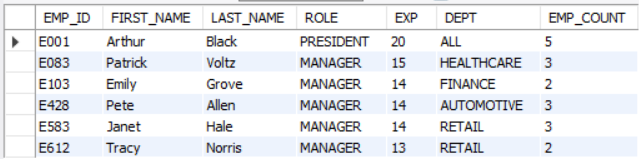
AND e.EMP\_ID != e.MANAGER\_ID

WHERE m.ROLE IN ("MANAGER", "PRESIDENT", "CEO")

GROUP BY m.EMP\_ID

ORDER BY m.EMP\_ID;

**Output:**



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

**SQL code:**

SELECT e.EMP\_ID as ID,

CONCAT(e.FIRST\_NAME,' ',e.LAST\_NAME) AS `NAME`,

e.DEPT

FROM employee.emp\_record\_table e

WHERE e.DEPT IN ("HEALTHCARE")

UNION

SELECT e.EMP\_ID as ID,

CONCAT(e.FIRST\_NAME,' ',e.LAST\_NAME) AS `NAME`,

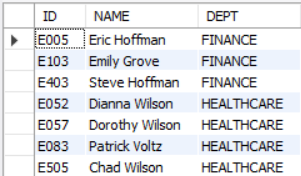
e.DEPT

FROM employee.emp\_record\_table e

WHERE e.DEPT IN ("FINANCE")

ORDER BY DEPT, ID;

**Output:**



1. 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.

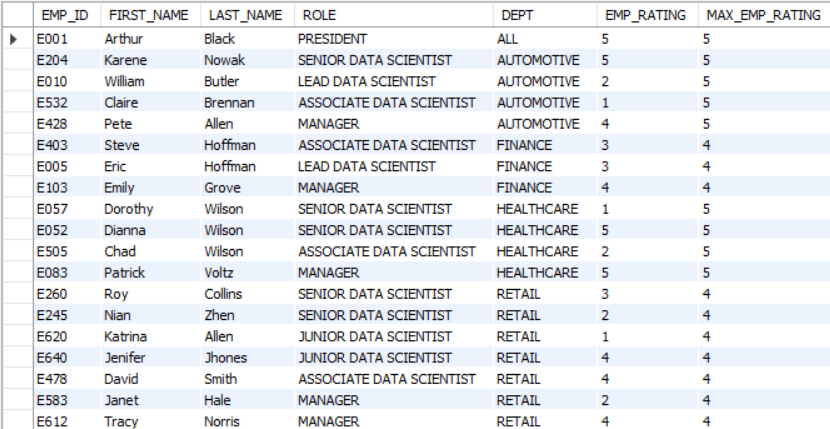
**SQL code:**

SELECT EMP\_ID, FIRST\_NAME, LAST\_NAME, ROLE, DEPT, EMP\_RATING, MAX(EMP\_RATING)

OVER (PARTITION BY DEPT) AS MAX\_EMP\_RATING

FROM employee.emp\_record\_table;

**Output:**

****

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

**SQL code:**

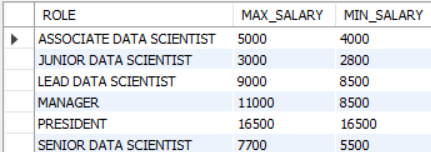
SELECT DISTINCT(ROLE), MAX(SALARY)

OVER (PARTITION BY ROLE) MAX\_SALARY, MIN(SALARY)

OVER (PARTITION BY ROLE) MIN\_SALARY

FROM employee.emp\_record\_table;

**Output:**



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

**SQL code:**

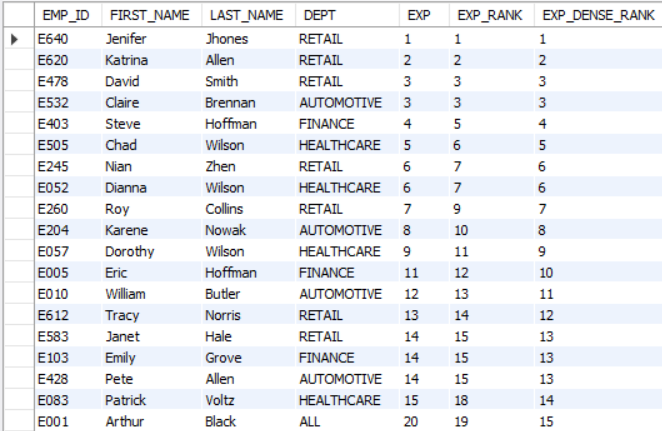
SELECT EMP\_ID, FIRST\_NAME, LAST\_NAME, DEPT, EXP,

RANK() OVER (ORDER BY EXP) EXP\_RANK,

DENSE\_RANK() OVER (ORDER BY EXP) EXP\_DENSE\_RANK

FROM employee.emp\_record\_table;

**Output:**

****

1. 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.

**SQL code:**

CREATE VIEW Employee\_View

AS

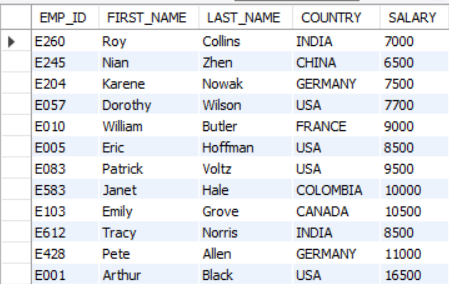
SELECT EMP\_ID, FIRST\_NAME, LAST\_NAME, COUNTRY, SALARY

FROM employee.emp\_record\_table

WHERE SALARY > 6000;

SELECT \* FROM Employee\_view;

**Output:**

****

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

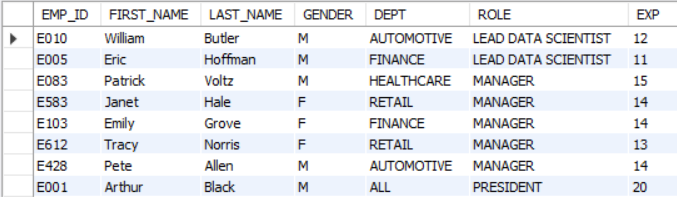
**SQL code:**

SELECT EMP\_ID, FIRST\_NAME, LAST\_NAME, GENDER, DEPT, ROLE, EXP

FROM employee.emp\_record\_table

WHERE ROLE IN (SELECT ROLE FROM employee.emp\_record\_table HAVING EXP >10);

**Output:**

****

1. 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.

**SQL code:**

DELIMITER &&

CREATE PROCEDURE experienced\_personal()

BEGIN

SELECT EMP\_ID, FIRST\_NAME, LAST\_NAME, GENDER, DEPT, ROLE, EXP

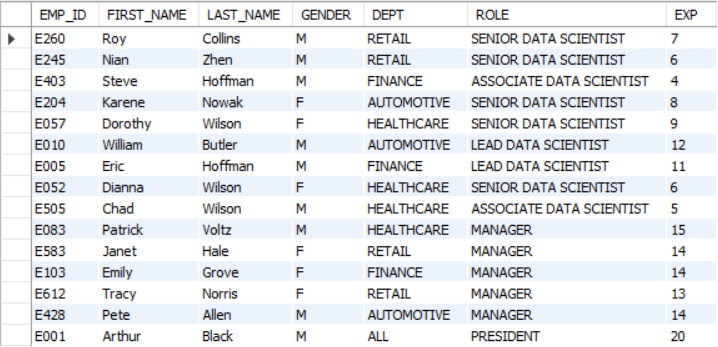
FROM employee.emp\_record\_table WHERE EXP > 3;

END &&

DELIMITER ;

call experienced\_personal();

**Output:**

****

1. 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'.

**SQL code:**

DELIMITER $$

drop FUNCTION Employee\_details;

CREATE FUNCTION Employee\_details(EXP int)

RETURNS VARCHAR(255) DETERMINISTIC

BEGIN DECLARE Employee\_details VARCHAR(255);

IF EXP <= 2 THEN SET Employee\_details = 'JUNIOR DATA SCIENTIST';

ELSEIF EXP <= 5 THEN SET Employee\_details = 'ASSOCIATE DATA SCIENTIST';

ELSEIF EXP <= 10 THEN SET Employee\_details = 'SENIOR DATA SCIENTIST';

ELSEIF EXP <= 12 THEN SET Employee\_details = 'LEAD DATA SCIENTIST';

ELSEIF EXP <= 16 THEN SET Employee\_details = 'MANAGER';

END IF;

RETURN (Employee\_details);

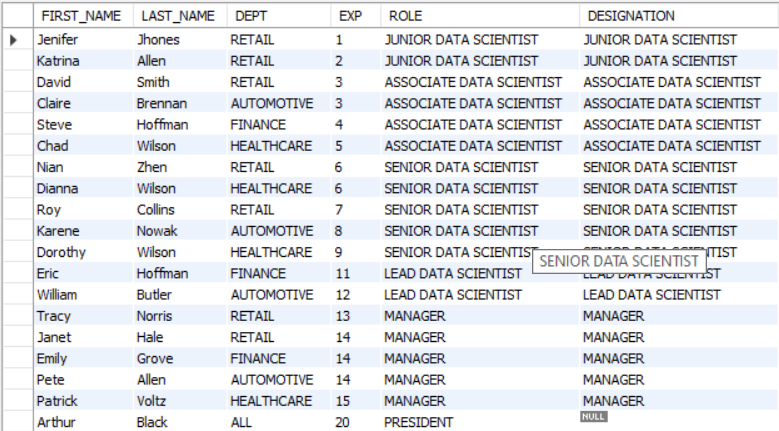
END$$

DELIMITER $$;

SELECT FIRST\_NAME, LAST\_NAME, DEPT, EXP, ROLE, Employee\_details(EXP) as DESIGNATION

FROM employee.emp\_record\_table ORDER BY EXP;

**Output:**



1. 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.

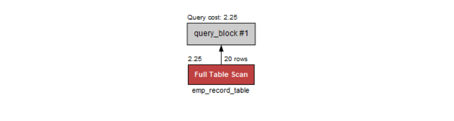
**SQL code: Before Indexing**

SELECT \*

FROM employee.emp\_record\_table

WHERE FIRST\_NAME='Eric';

**Output:**



**SQL code: Indexing**

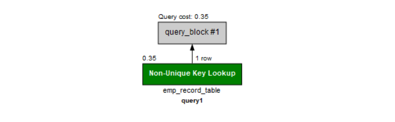
CREATE INDEX query1 ON employee.emp\_record\_table(FIRST\_NAME);

SELECT \*

FROM employee.emp\_record\_table

WHERE FIRST\_NAME='Eric';

**Output:**



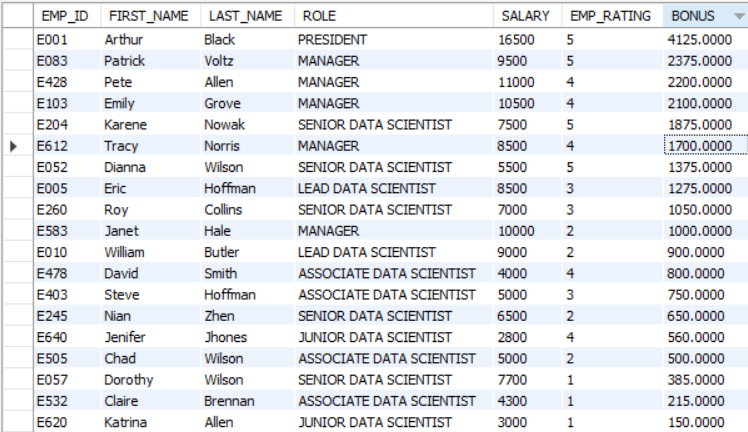
1. Write a query to calculate the bonus for all the employees, based on their ratings and salaries (Use the formula: 5% of salary \* employee rating).

**SQL code:**

SELECT EMP\_ID, FIRST\_NAME, LAST\_NAME, ROLE, SALARY, EMP\_RATING, (SALARY/20)\*EMP\_RATING AS BONUS

FROM employee.emp\_record\_table;

**Output:**



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

**SQL code:**

SELECT CONTINENT, COUNTRY, AVG(SALARY) as AVERAGE\_SALARY

FROM employee.emp\_record\_table

GROUP BY CONTINENT, COUNTRY WITH ROLLUP;

**Output:**

