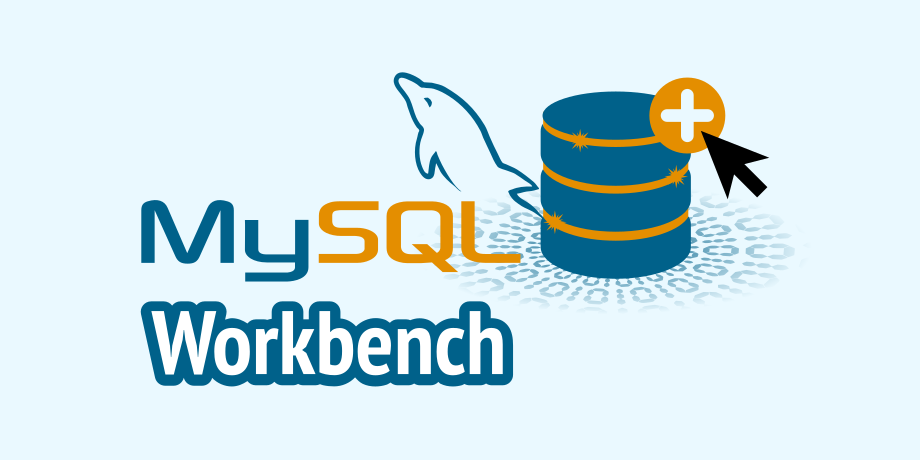
SQL Project : Science Qtech Employee Performance Mapping

Kaushik Dey



**DESCRIPTION :**

ScienceQtech is a startup that works in the Data Science field. ScienceQtech has worked on fraud detection, market basket, self-driving cars, supply chain, algorithmic early detection of lung cancer, customer sentiment, and the drug discovery field. With the annual appraisal cycle around the corner, the HR department has asked you (Junior Database Administrator) to generate reports on employee details, their performance, and on the project that the employees have undertaken, to analyze the employee database and extract specific data based on different requirements.

**Objective:**

To facilitate a better understanding, managers have provided ratings for each employee which will help the HR department to finalize the employee performance mapping. As a DBA, you should find the maximum salary of the employees and ensure that all jobs are meeting the organization's profile standard. You also need to calculate bonuses to find extra cost for expenses. This will raise the overall performance of the organization by ensuring that all required employees receive training.

**Dataset description :**

**emp\_record\_table** : It contains the information of all the employees.

* EMP\_ID – ID of the employee
* FIRST\_NAME – First name of the employee
* LAST\_NAME – Last name of the employee
* GENDER – Gender of the employee
* ROLE – Post of the employee
* DEPT – Field of the employee
* EXP – Years of experience the employee has
* COUNTRY – Country in which the employee is presently living
* CONTINENT – Continent in which the country is
* SALARY – Salary of the employee
* EMP\_RATING – Performance rating of the employee
* MANAGER\_ID – The manager under which the employee is assigned
* PROJ\_ID – The project on which the employee is working or has worked on

**Proj\_table** : It contains information about the projects .

* PROJECT\_ID – ID for the project
* PROJ\_Name – Name of the project
* DOMAIN – Field of the project
* START\_DATE – Day the project began
* CLOSURE\_DATE – Day the project was or will be completed
* DEV\_QTR – Quarter in which the project was scheduled
* STATUS – Status of the project currentl

**Data\_science\_team** : It contains information about all the employees in the Data Science team.

* EMP\_ID – ID of the employee
* FIRST\_NAME – First name of the employee
* LAST\_NAME – Last name of the employee
* GENDER – Gender of the employee
* ROLE – Post of the employee
* DEPT – Field of the employee
* EXP – Years of experience the employee has
* COUNTRY – Country in which the employee is presently living
* CONTINENT – Continent in which the country is

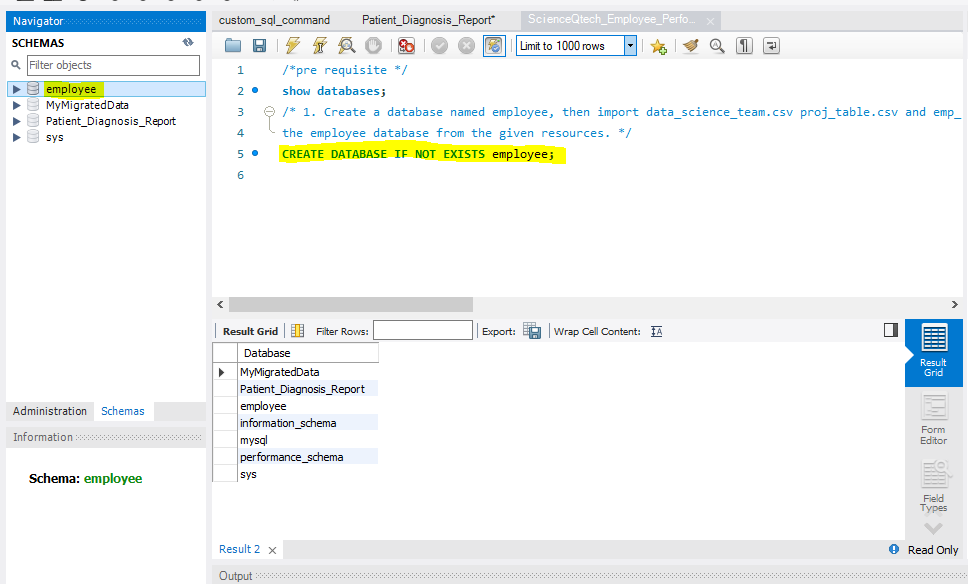
The tasks to be performed:

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.

**SQL CODE**

CREATE DATABASE employee;

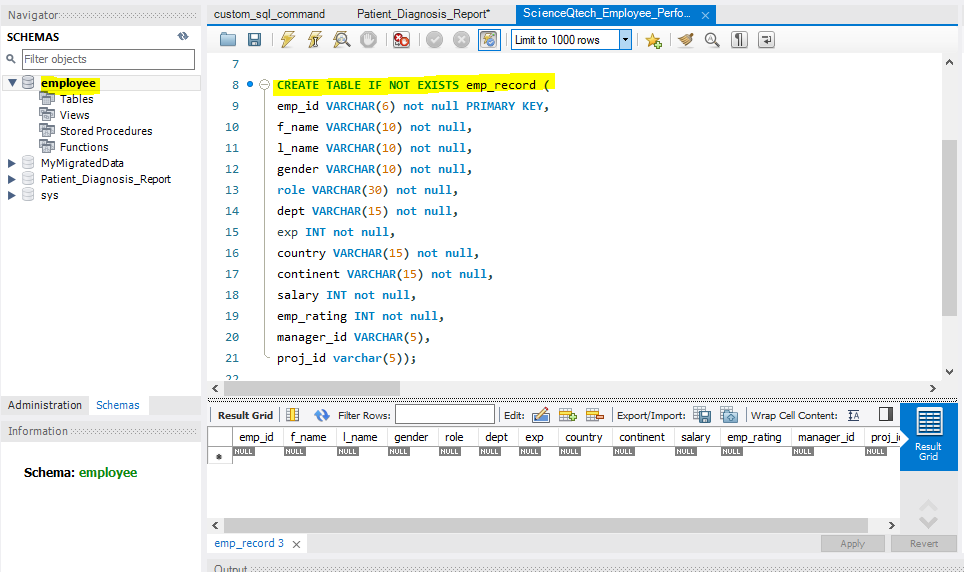
**OUTPUT :**



**SQL CODE**

CREATE TABLE IF NOT EXISTS 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));

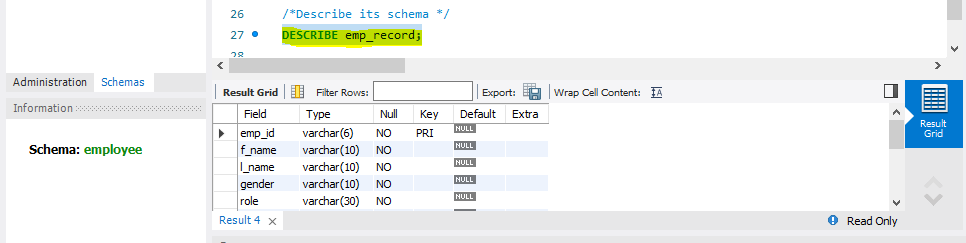
**OUTPUT**



**SQL CODE**

DESCRIBE emp\_record;

**OUTPUT**



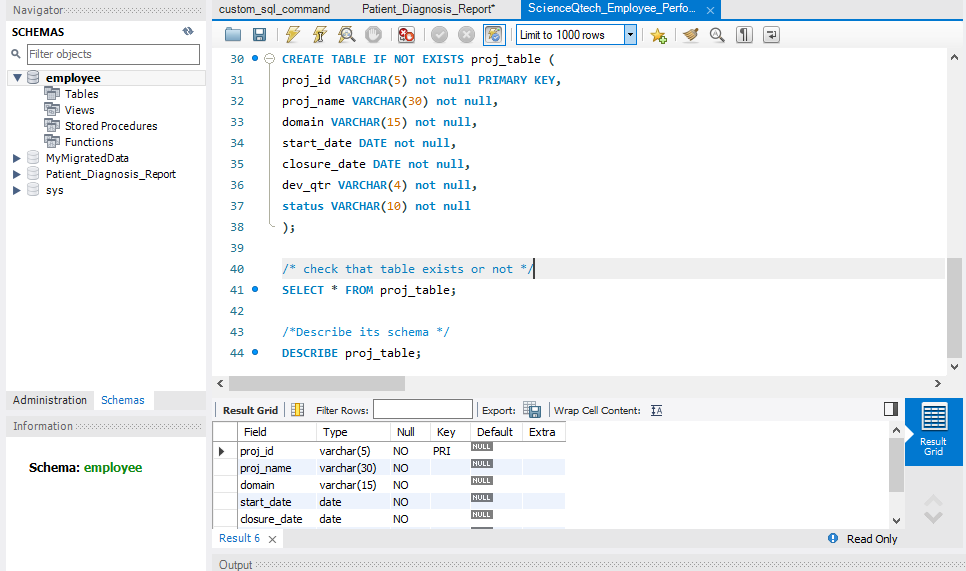
**SQL CODE**

CREATE TABLE IF NOT EXISTS 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);

SELECT \* FROM proj\_table;

DESCRIBE proj\_table;

**OUTPUT**



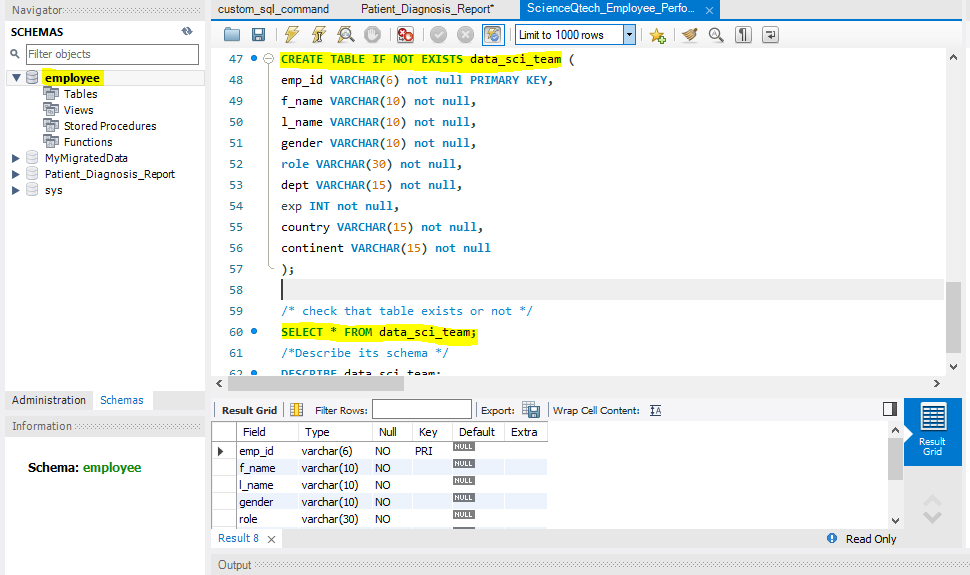
**SQL CODE**

CREATE TABLE IF NOT EXISTS 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);

SELECT \* FROM data\_sci\_team;

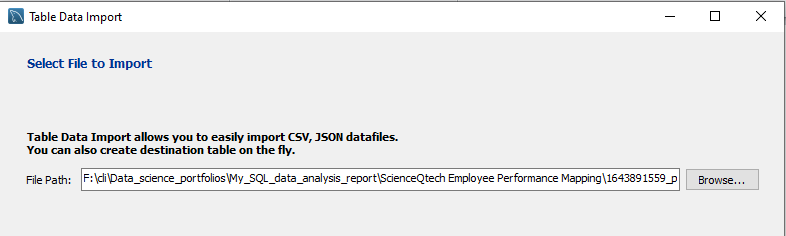
DESCRIBE data\_sci\_team;

**OUTPUT**

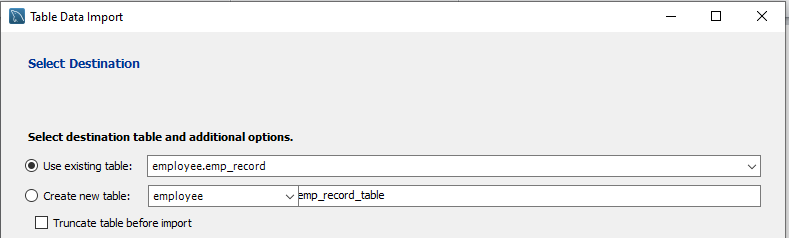


**IMPORT DATA INTO TABLES (First approach normally UI based)**

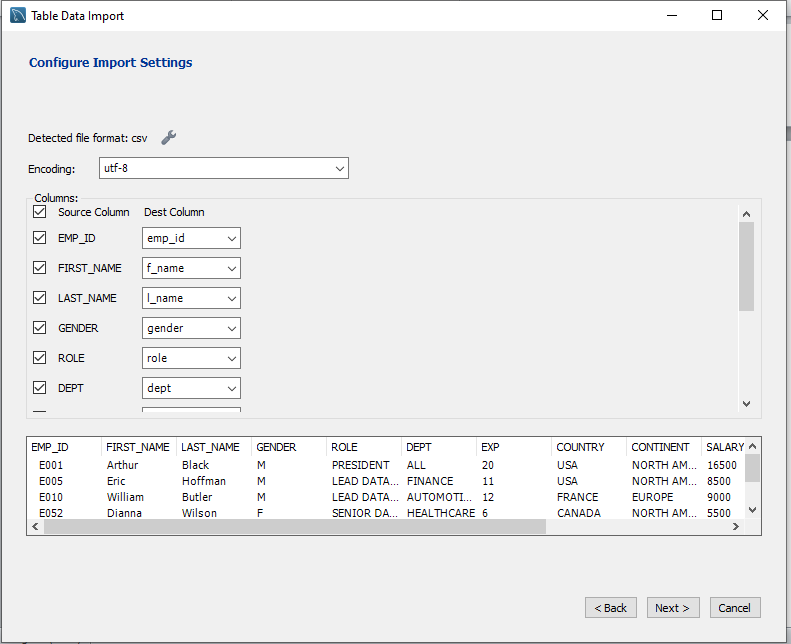
1. **First right click on the table and select Table Data Import wizard.**
2. **Next Browse the file path.**



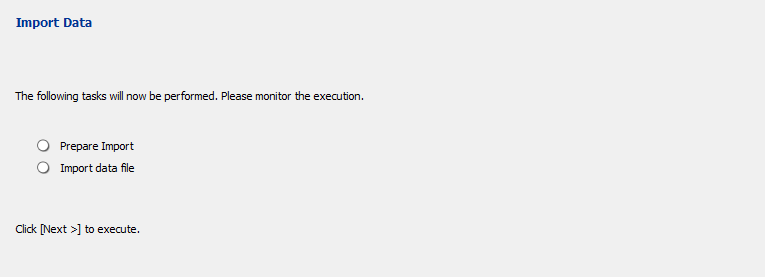
1. **Click on next and select destination.**



1. **Configuare import settings**



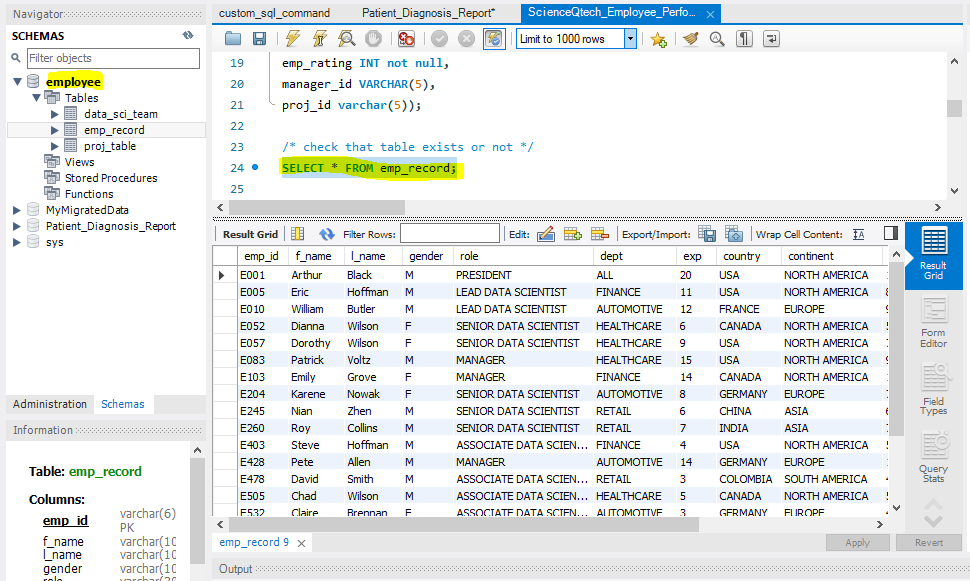
1. **Click on next, and finally import data.**



1. **When task done then we can verify the data is present or not.**

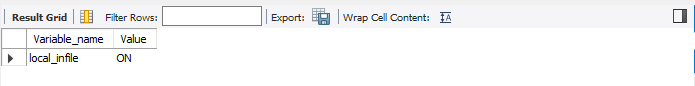
SELECT \* FROM emp\_record;

**emp\_record**



**IMPORT DATA INTO TABLES (via command line)**

SHOW GLOBAL VARIABLES LIKE 'local\_infile';



LOAD DATA LOCAL INFILE 'F:\cli\Data\_science\_portfolios\My\_SQL\_data\_analysis\_report\ScienceQtech Employee Performance Mapping\1643891559\_performance\_mapping\_datasets\proj\_table.csv'

INTO TABLE employee.proj\_table

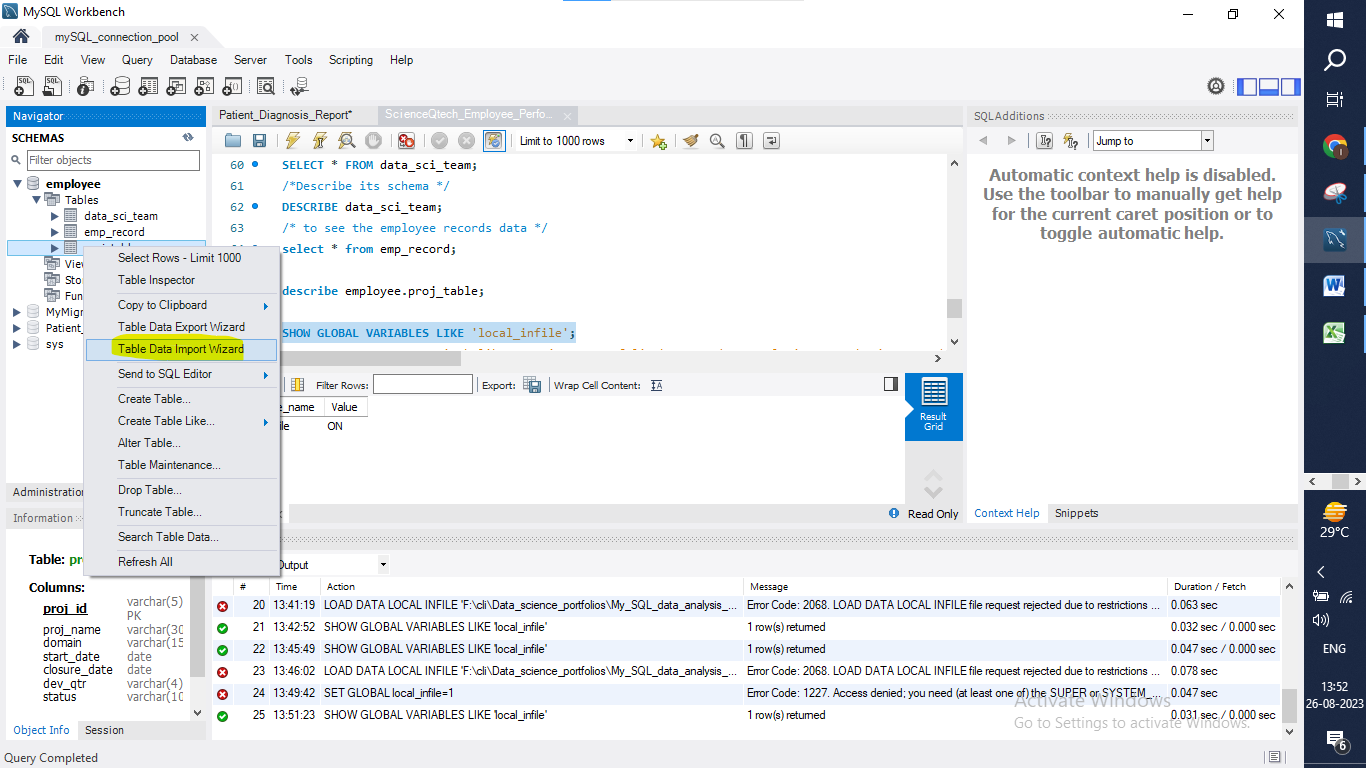
FIELDS TERMINATED BY ','

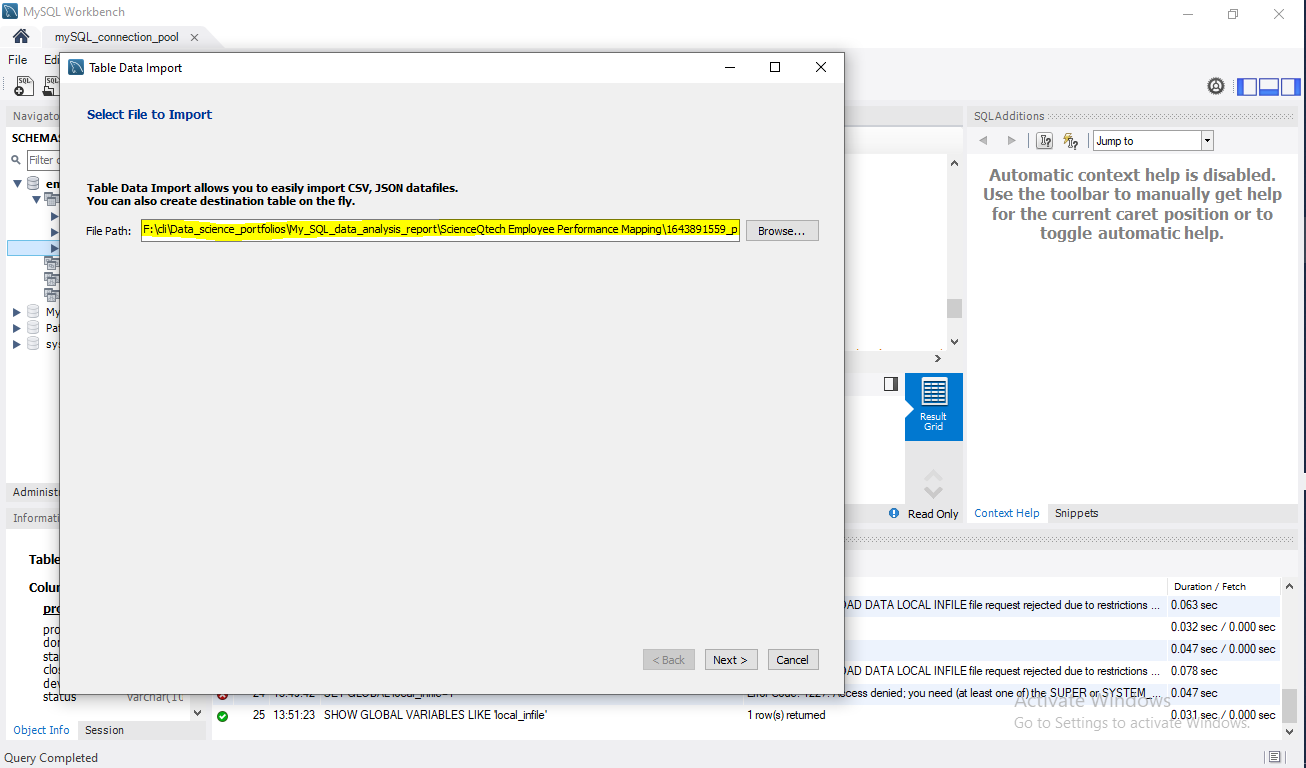
ENCLOSED BY '"' LINES

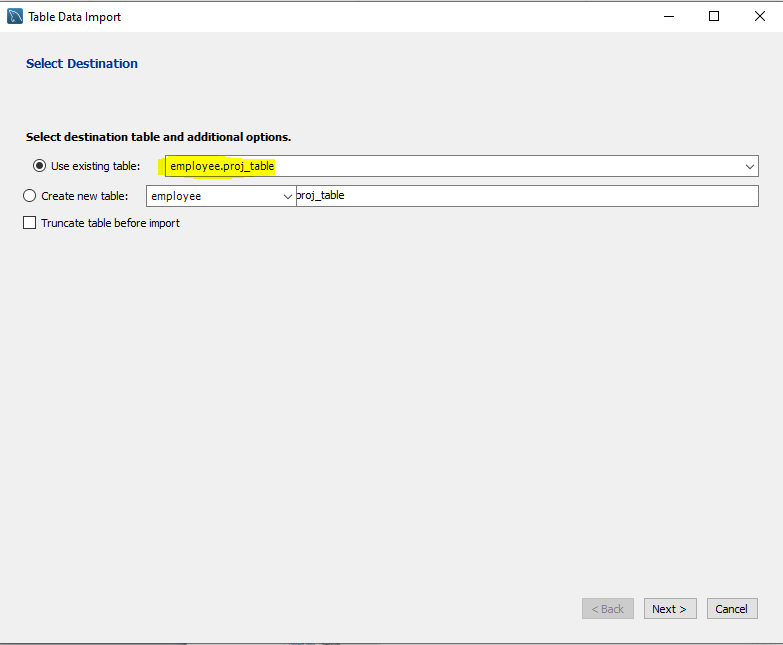
TERMINATED BY '\n' IGNORE 1 ROWS

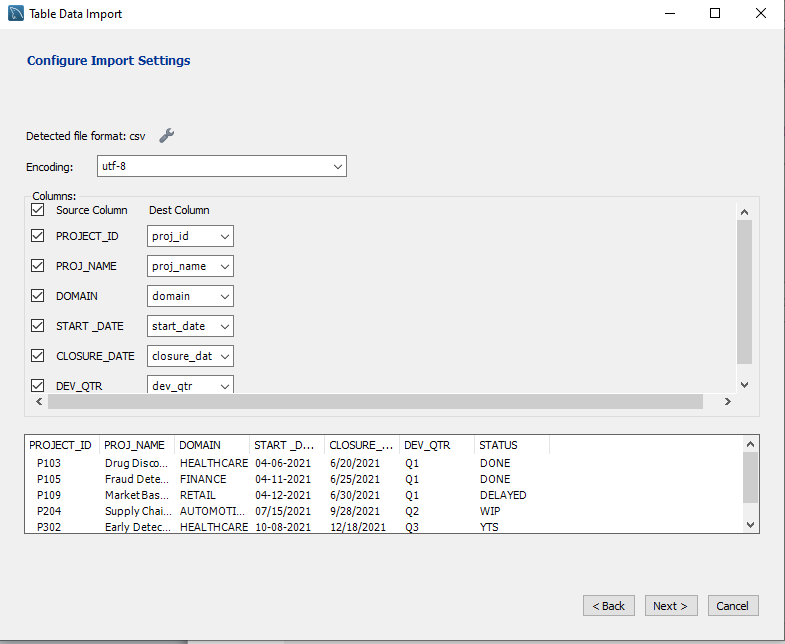
(`proj\_id`,`proj\_name`,`domain`,`start\_date`,`closure\_date`,`dev\_qtr`,`status`);

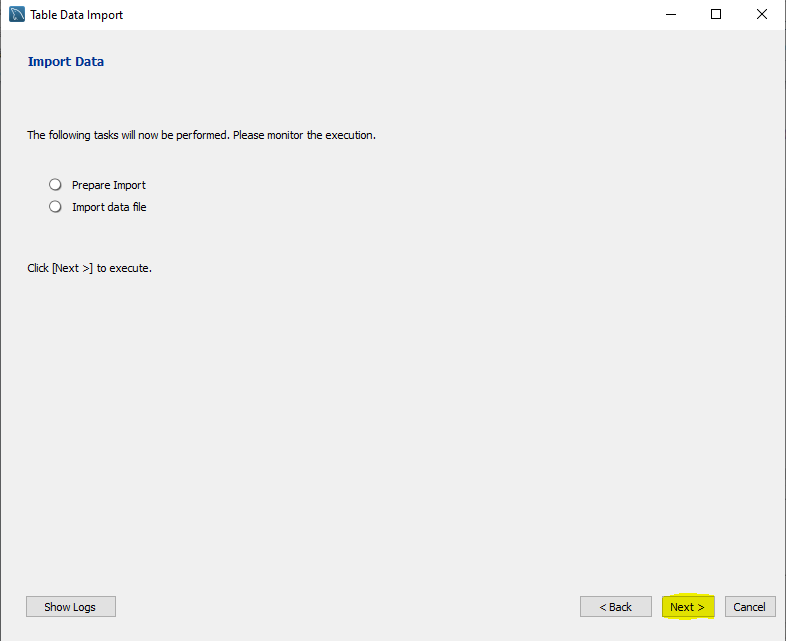
**Otherwise, we have to import the data in same manner as follows in step 1 and the screenshots are given below.**

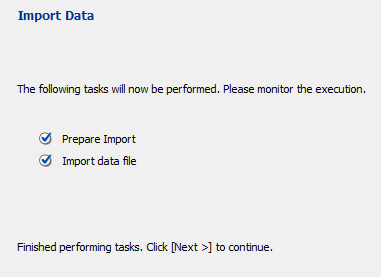
****

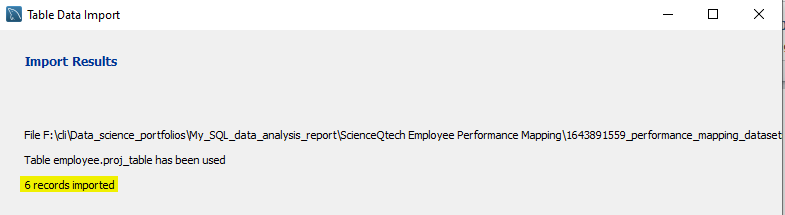








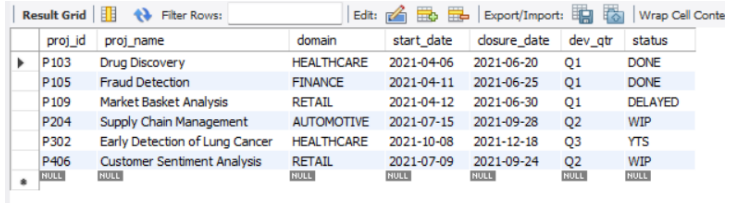




**And now if we check the table with the following commands it can show the results .**

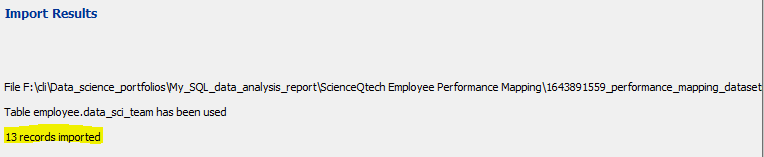
**proj\_table**

**select \* from employee.proj\_table;**

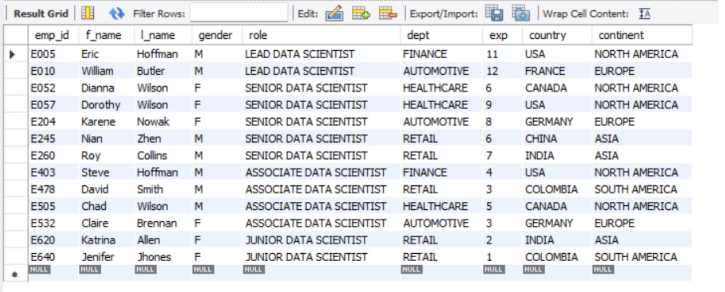


**Data\_science\_team**

Same step we have to follow and the result is given below screenshots.



**select \* from employee.data\_sci\_team;**

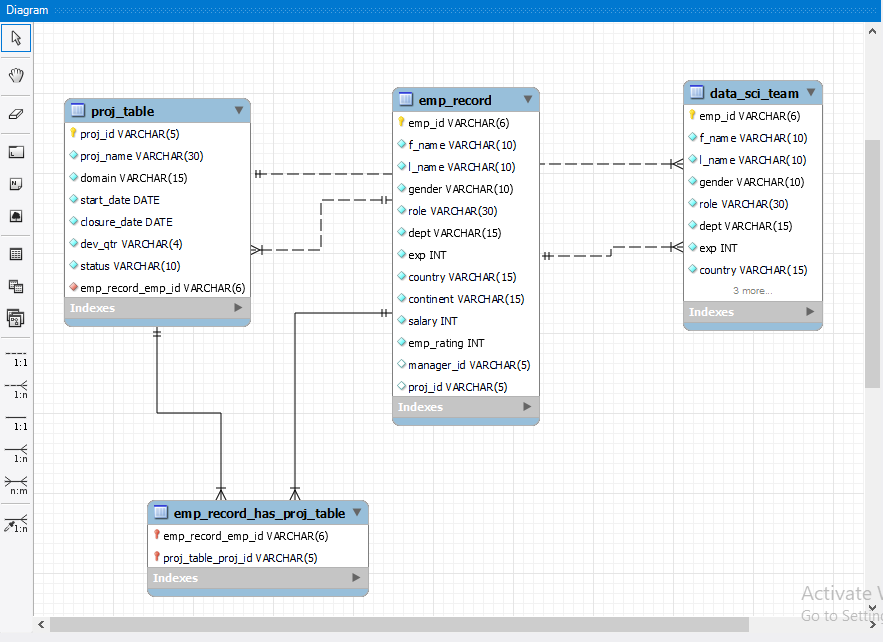


1. Create an ER diagram for the given employee database.

**SQL code:**

Reverse Engineering of employee database.

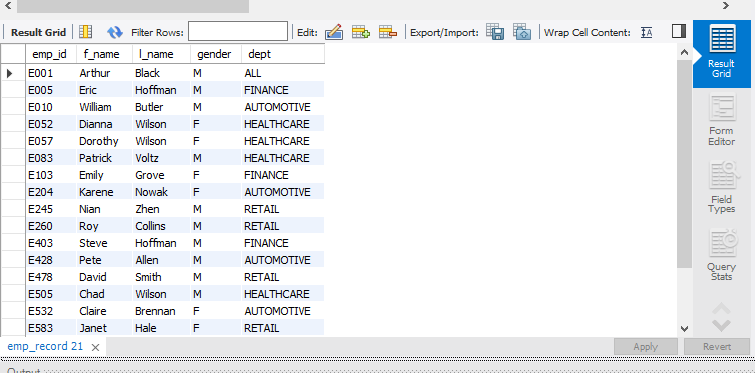
**Output:**



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, f\_name, l\_name, gender, dept from employee.emp\_record;

**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

**Less than two:**

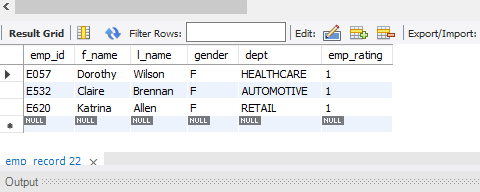
**SQL CODE :**

select emp\_id, f\_name, l\_name, gender, dept, emp\_rating

from employee.emp\_record

where emp\_rating < 2;

**Output:**



**Greater than four:**

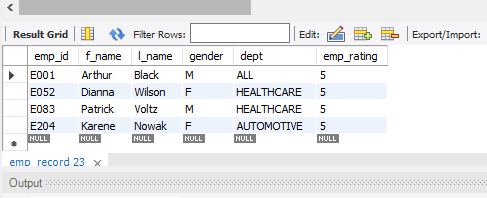
**SQL CODE :**

SELECT emp\_id, f\_name, l\_name, gender, dept, emp\_rating

FROM employee.emp\_record

WHERE emp\_rating > 4;

**Output:**



**Between two and four:**

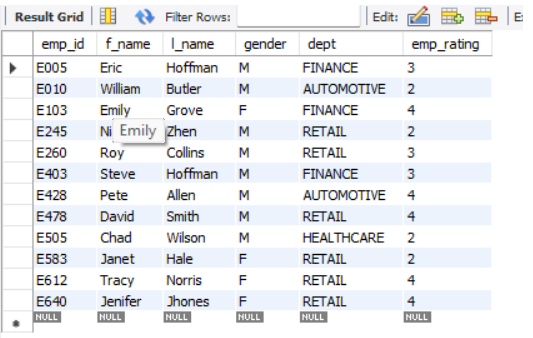
**SQL CODE :**

SELECT emp\_id, f\_name, l\_name, gender, dept, emp\_rating

FROM employee.emp\_record

WHERE emp\_rating between 2 and 4;

**Output:**



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.

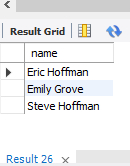
**SQL CODE :**

select CONCAT(f\_name,' ',l\_name) name

from employee.emp\_record

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 :**

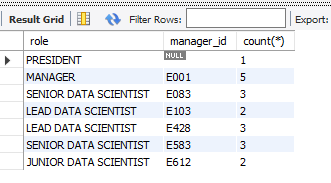
select role, manager\_id, count(\*)

from employee.emp\_record

group by manager\_id

order by manager\_id;

**Output:**



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

**SQL CODE :**

SELECT f\_name, l\_name, dept

FROM employee.emp\_record

WHERE dept = 'HEALTHCARE'

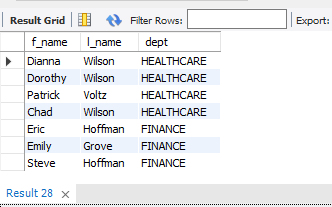
UNION

SELECT f\_name, l\_name, dept

FROM emp\_record

WHERE dept = 'FINANCE';

**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, f\_name, l\_name, role, dept, emp\_rating, emp\_rating AS max\_rating

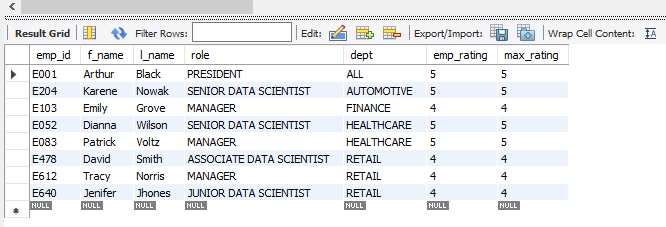
FROM emp\_record

WHERE (dept, emp\_rating)

IN (SELECT dept, MAX(emp\_rating) FROM emp\_record GROUP By dept)

ORDER BY dept ASC;

**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.

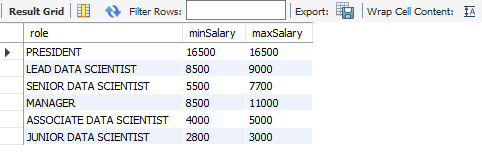
**SQL CODE :**

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

FROM employee.emp\_record

GROUP BY role;

**Output :**



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

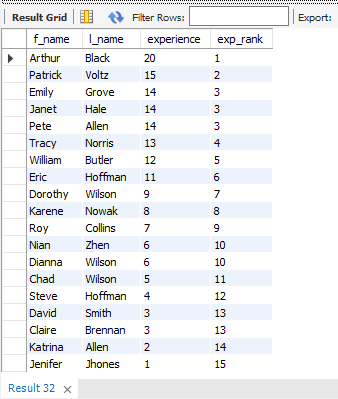
**SQL CODE :**

SELECT f\_name, l\_name, exp as experience,

DENSE\_RANK() OVER (ORDER BY exp DESC) exp\_rank

FROM employee.emp\_record;

**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 6K\_salary\_max AS

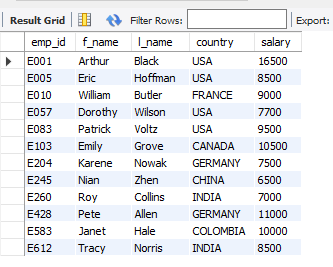
SELECT emp\_id, f\_name, l\_name, country, salary

FROM employee.emp\_record

WHERE salary > 6000;

SELECT \* FROM 6K\_salary\_max;

**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, f\_name, l\_name, exp

FROM employee.emp\_record

WHERE exp IN (

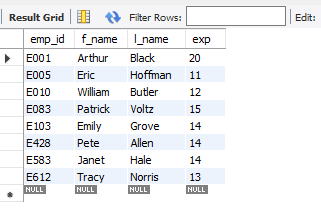
SELECT exp

FROM emp\_record

WHERE 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 Employee3()

BEGIN

SELECT \* FROM employee.emp\_record

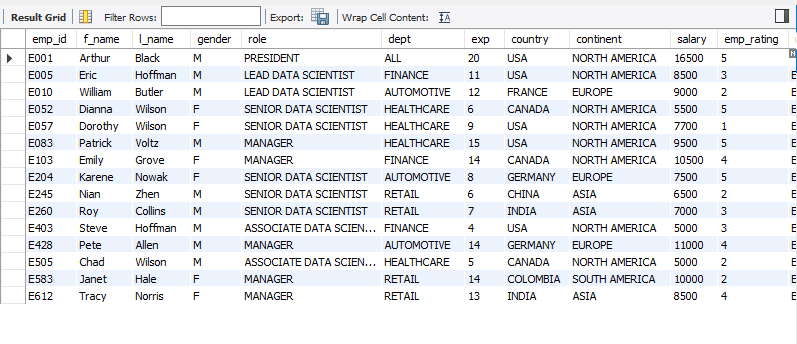
WHERE exp > 3;

END //

DELIMITER ;

call Employee3();

**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'.

CREATE PROCEDURE `check\_role` ()

BEGIN

SELECT \* ,

CASE

WHEN exp <= 2 THEN role = 'JUNIOR DATA SCIENTIST'

WHEN exp BETWEEN 3 AND 5 THEN role = 'ASSOCIATE DATA SCIENTIST'

WHEN exp BETWEEN 6 AND 10 THEN role = 'SENIOR DATA SCIENTIST'

WHEN exp BETWEEN 11 AND 12 THEN role = 'LEAD DATA SCIENTIST'

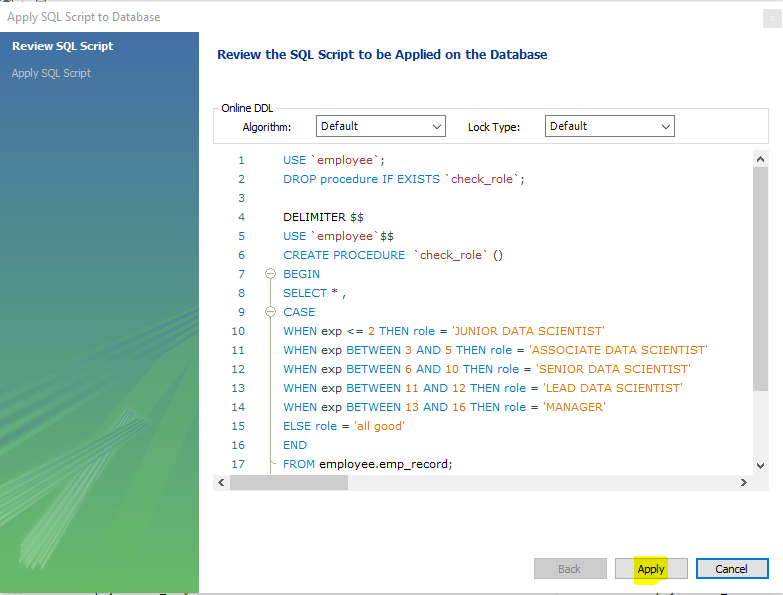
WHEN exp BETWEEN 13 AND 16 THEN role = 'MANAGER'

ELSE role = 'all good'

END

FROM employee.emp\_record;

END



**Alter Store Procedure**

CREATE DEFINER=`admin`@`%` PROCEDURE `check\_role`()

BEGIN

SELECT \* ,

CASE

WHEN exp <= 2 THEN role = 'JUNIOR DATA SCIENTIST'

WHEN exp BETWEEN 3 AND 5 THEN role = 'ASSOCIATE DATA SCIENTIST'

WHEN exp BETWEEN 6 AND 10 THEN role = 'SENIOR DATA SCIENTIST'

WHEN exp BETWEEN 11 AND 12 THEN role = 'LEAD DATA SCIENTIST'

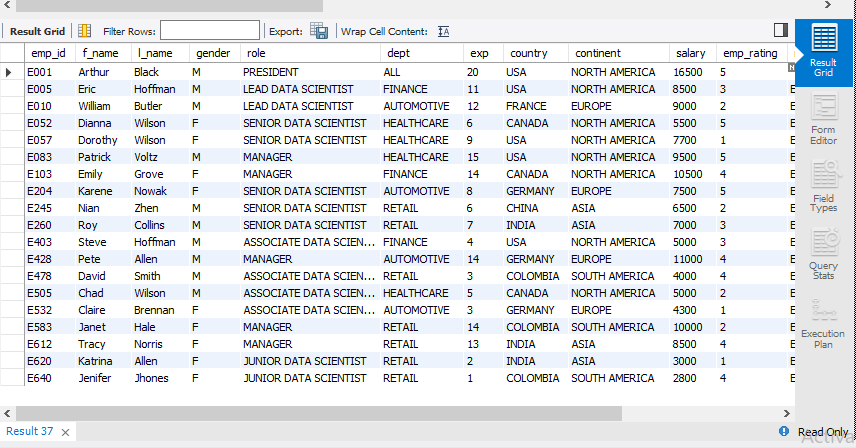
WHEN exp BETWEEN 13 AND 16 THEN role = 'MANAGER'

ELSE role = 'all good'

END AS QuantityText

FROM employee.emp\_record;

END



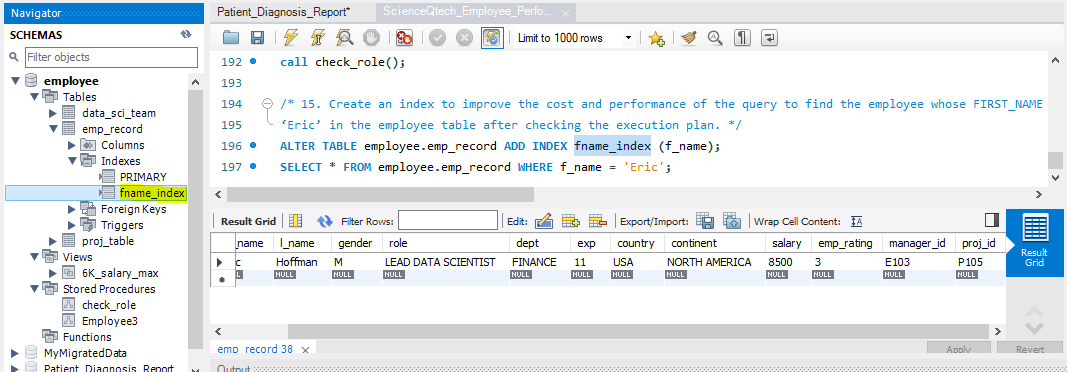
1. Create an index to improve the cost and performance of the query to find the employee whose FIRST\_NAME is ‘Eric’ in the employee table after checking the execution plan.

**SQL code:**

ALTER TABLE employee.emp\_record ADD INDEX fname\_index (f\_name);

SELECT \* FROM emp\_record WHERE f\_name = ‘Eric’;

**output:**



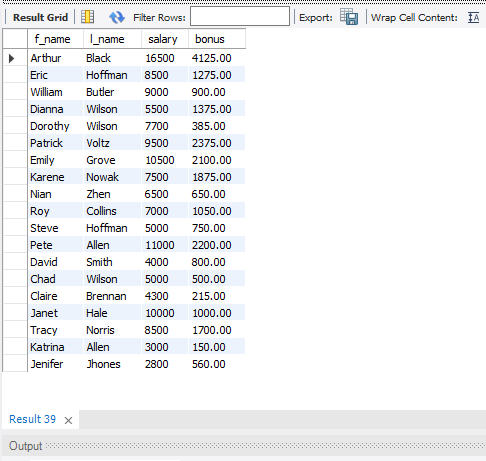
1. Write a query to calculate the bonus for all the employees, based on their ratings and salaries (Use the formula: 5% of salary \* employee rating).

**SQL code:**

SELECT f\_name, l\_name, salary, ((salary \* .05)\*emp\_rating) AS bonus

FROM employee.emp\_record;

**Output:**



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

Average salary based on the continent.

**SQL code:**

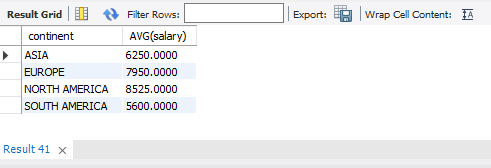
SELECT continent, AVG(salary)

FROM employee.emp\_record

GROUP BY continent

ORDER BY continent ASC;

**Output:**



Average salary based on the country.

**SQL code:**

SELECT country, AVG(salary)

FROM employee.emp\_record

GROUP BY country

ORDER BY country ASC;

**Output:**

