

SQL Queries

Project 1

By Nora Alsaeed

ScienceQtech Employee Performance Mapping

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.

Note: You must download the dataset from the course resource section in LMS and create a table to perform the above objective.

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 currently

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 task to be performed:

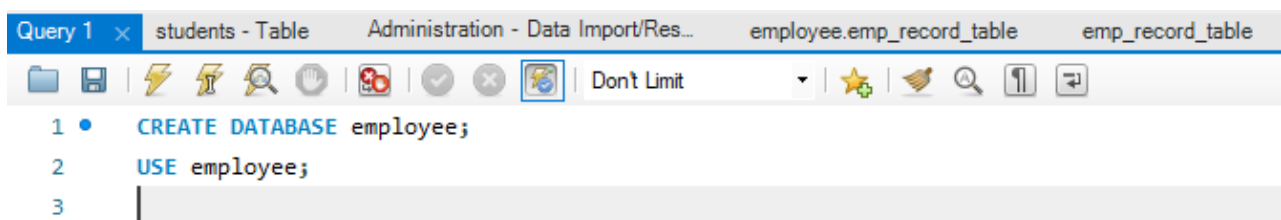
1. Create a database named employee, then import data_science_team.csv proj_table.csv and emp_record_table.csv into the employee database from the given resources.

SQL code:

```
CREATE DATABASE employee;
```

```
USE employee;
```

Output:



Output			
Action Output			
#	Time	Action	Message
✓ 1	21:04:09	CREATE DATABASE employee	1 row(s) affected
✓ 2	21:04:11	USE employee	0 row(s) affected

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

SQL code:

```
DESCRIBE data_science_team;
```

```
DESCRIBE emp_record_table;
```

```
DESCRIBE proj_table;
```

```
ALTER TABLE emp_record_table
```

```
ADD primary key (EMP_ID);
```

```
ALTER TABLE emp_record_table
```

```
MODIFY EMP_ID VARCHAR(30);
```

```
ALTER TABLE data_science_team
```

```
MODIFY EMP_ID VARCHAR(30);
```

```
ALTER TABLE data_science_team
```

```
MODIFY EMP_ID VARCHAR(30);
```

```
ALTER TABLE proj_table
```

```
MODIFY PROJECT_ID VARCHAR(50);
```

```
ALTER TABLE emp_record_table
```

```
MODIFY PROJ_ID VARCHAR(50);
```

```
ALTER TABLE proj_table
```

```
ADD primary key (PROJECT_ID);
```

```
ALTER TABLE emp_record_table
```

```
ADD foreign key (PROJ_ID)
```

```
REFERENCES proj_table(PROJECT_ID);
```

```
ALTER TABLE emp_record_table
```

```
ADD FOREIGN KEY(PROJ_ID)
```

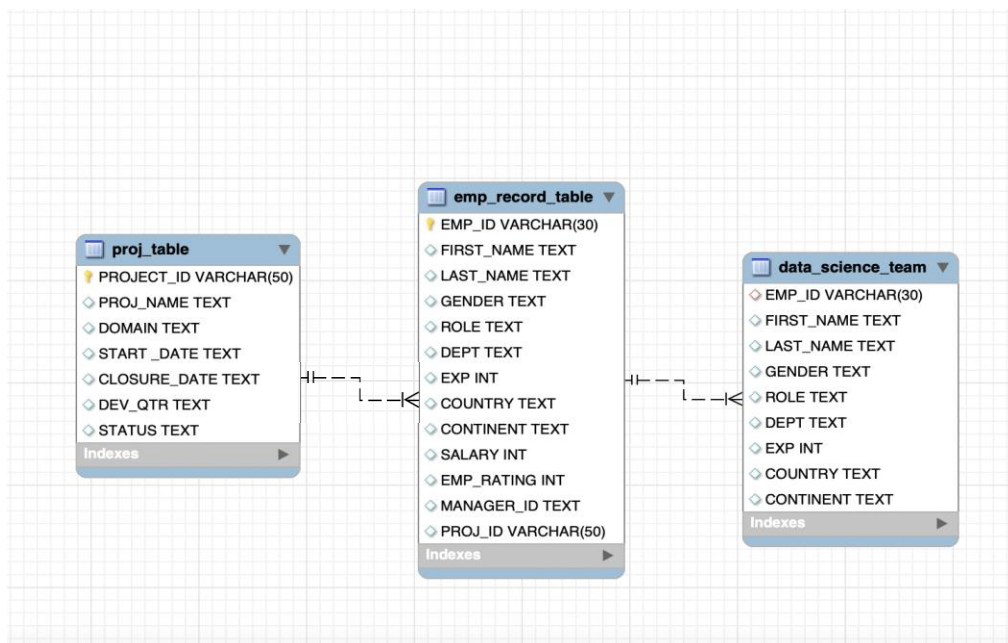
```
REFERENCES proj_table(PROJECT_ID);
```

```
ALTER TABLE data_science_team
```

```
ADD FOREIGN KEY(EMP_ID)
```

```
REFERENCES emp_record_table(EMP_ID);
```

Output:

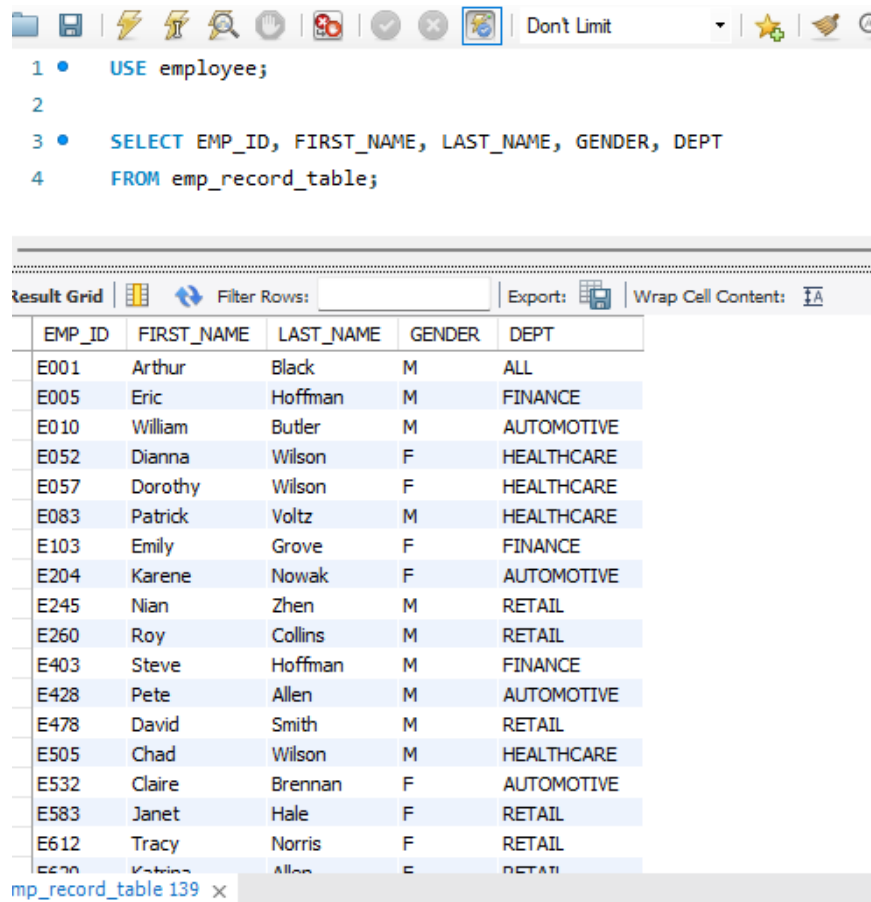


3. Write a query to fetch EMP_ID, FIRST_NAME, LAST_NAME, GENDER, and DEPARTMENT from the employee record table, and make a list of employees and details of their department.

SQL code:

```
SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT  
FROM emp_record_table;
```

Output:



The screenshot shows a SQL IDE interface. At the top, there is a toolbar with various icons. Below the toolbar, the SQL query is entered in a text area. The query is as follows:

```
1 • USE employee;  
2  
3 • SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT  
4 FROM emp_record_table;
```

Below the query, there is a "result Grid" section. It contains a table with the following data:

EMP_ID	FIRST_NAME	LAST_NAME	GENDER	DEPT
E001	Arthur	Black	M	ALL
E005	Eric	Hoffman	M	FINANCE
E010	William	Butler	M	AUTOMOTIVE
E052	Dianna	Wilson	F	HEALTHCARE
E057	Dorothy	Wilson	F	HEALTHCARE
E083	Patrick	Voltz	M	HEALTHCARE
E103	Emily	Grove	F	FINANCE
E204	Karene	Nowak	F	AUTOMOTIVE
E245	Nian	Zhen	M	RETAIL
E260	Roy	Collins	M	RETAIL
E403	Steve	Hoffman	M	FINANCE
E428	Pete	Allen	M	AUTOMOTIVE
E478	David	Smith	M	RETAIL
E505	Chad	Wilson	M	HEALTHCARE
E532	Claire	Brennan	F	AUTOMOTIVE
E583	Janet	Hale	F	RETAIL
E612	Tracy	Norris	F	RETAIL
E620	Katrina	Allen	F	RETAIL

At the bottom of the screenshot, there is a tab labeled "emp_record_table 139" with a close button (X).

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

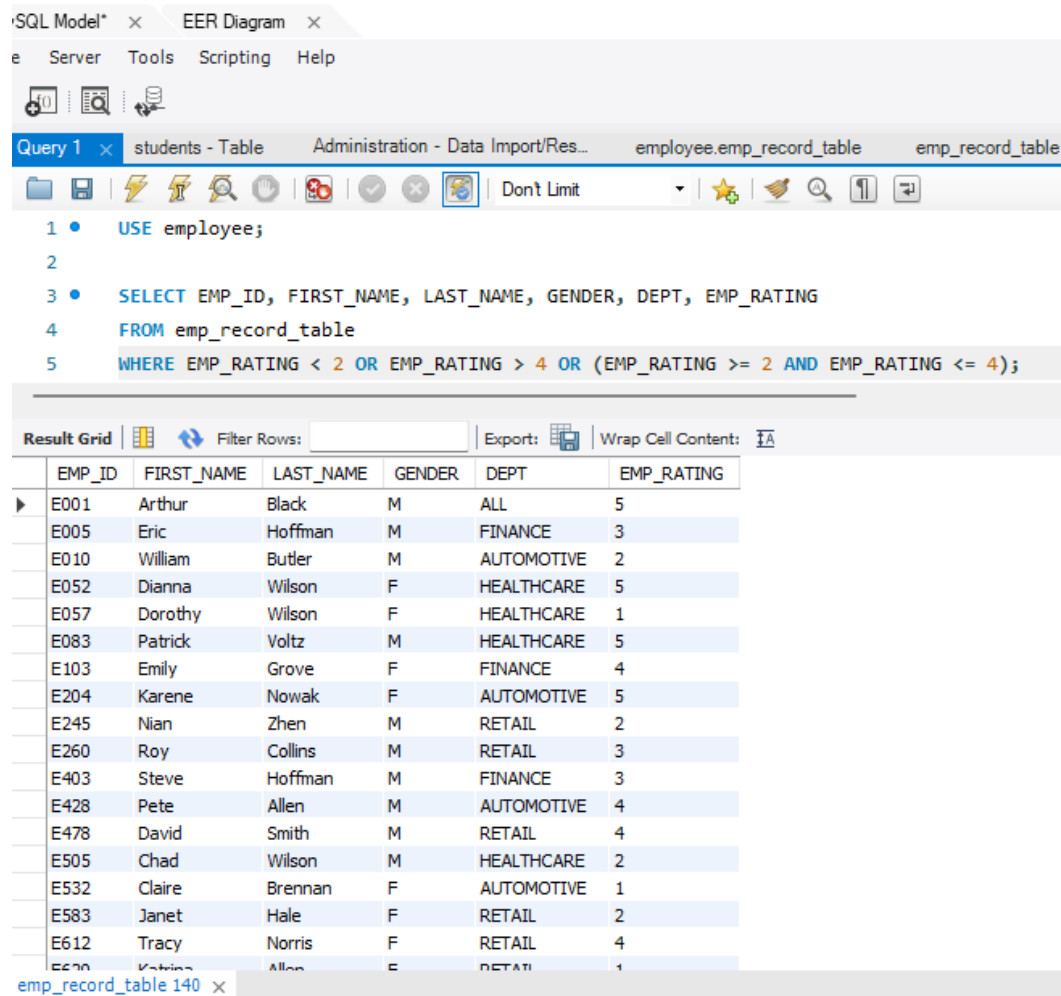
SQL code:

```
SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT, EMP_RATING
```

```
FROM emp_record_table
```

```
WHERE EMP_RATING < 2 OR EMP_RATING > 4 OR (EMP_RATING >= 2 AND EMP_RATING <= 4);
```

Output:



The screenshot shows the SQL Developer interface. The top pane displays the following SQL query:

```
1 • USE employee;  
2  
3 • SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT, EMP_RATING  
4 FROM emp_record_table  
5 WHERE EMP_RATING < 2 OR EMP_RATING > 4 OR (EMP_RATING >= 2 AND EMP_RATING <= 4);
```

The bottom pane shows the 'Result Grid' with the following data:

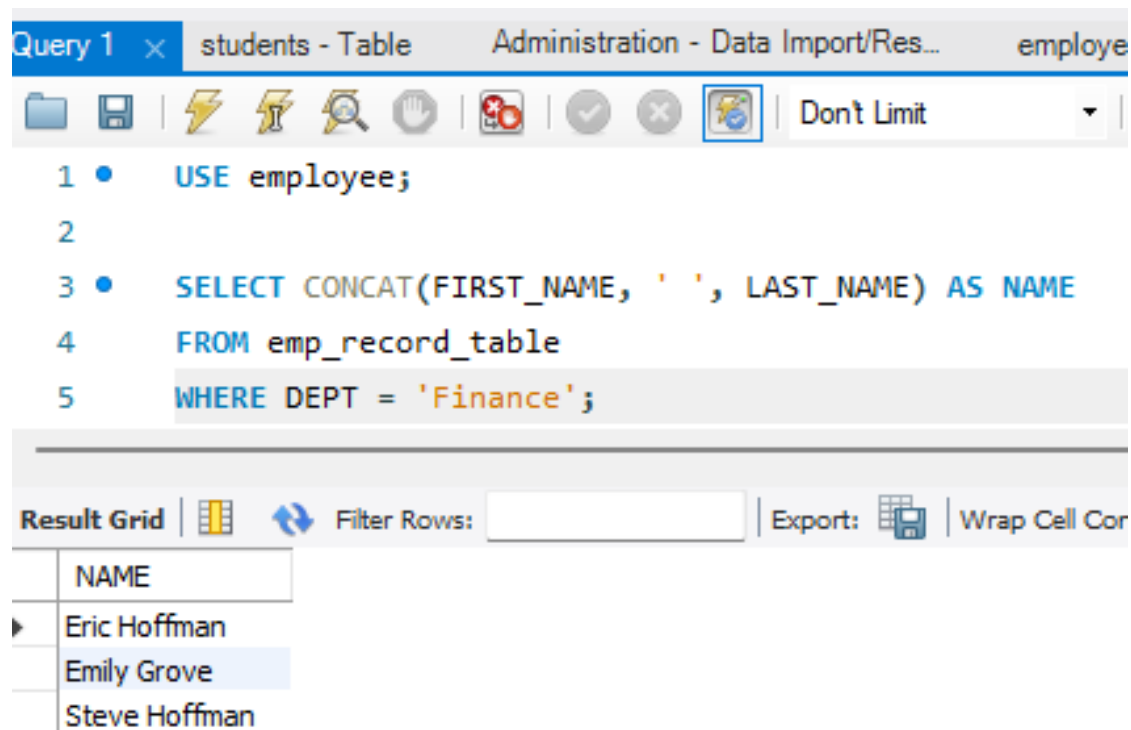
EMP_ID	FIRST_NAME	LAST_NAME	GENDER	DEPT	EMP_RATING
E001	Arthur	Black	M	ALL	5
E005	Eric	Hoffman	M	FINANCE	3
E010	William	Butler	M	AUTOMOTIVE	2
E052	Dianna	Wilson	F	HEALTHCARE	5
E057	Dorothy	Wilson	F	HEALTHCARE	1
E083	Patrick	Voltz	M	HEALTHCARE	5
E103	Emily	Grove	F	FINANCE	4
E204	Karene	Nowak	F	AUTOMOTIVE	5
E245	Nian	Zhen	M	RETAIL	2
E260	Roy	Collins	M	RETAIL	3
E403	Steve	Hoffman	M	FINANCE	3
E428	Pete	Allen	M	AUTOMOTIVE	4
E478	David	Smith	M	RETAIL	4
E505	Chad	Wilson	M	HEALTHCARE	2
E532	Claire	Brennan	F	AUTOMOTIVE	1
E583	Janet	Hale	F	RETAIL	2
E612	Tracy	Norris	F	RETAIL	4
E690	Yukina	Allen	F	RETAIL	1

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.

SQL code:

```
SELECT CONCAT(FIRST_NAME, ' ', LAST_NAME) AS NAME  
FROM emp_record_table  
WHERE DEPT = 'Finance';
```

Output:



The screenshot shows a SQL IDE interface. The top pane displays a query named 'Query 1' with the following SQL code:

```
1 • USE employee;  
2  
3 • SELECT CONCAT(FIRST_NAME, ' ', LAST_NAME) AS NAME  
4 FROM emp_record_table  
5 WHERE DEPT = 'Finance';
```

The bottom pane shows the 'Result Grid' with the following data:

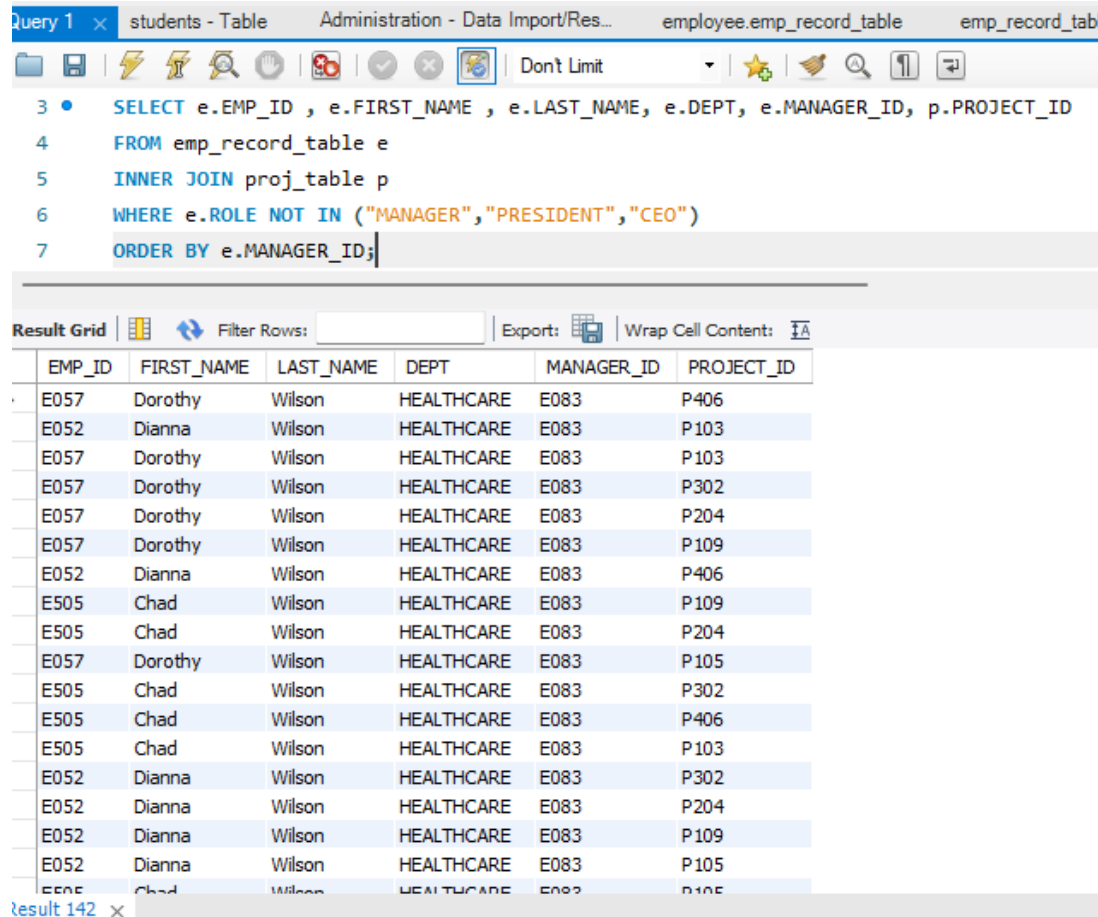
NAME
Eric Hoffman
Emily Grove
Steve Hoffman

6. 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 e.EMP_ID , e.FIRST_NAME , e.LAST_NAME, e.DEPT, e.MANAGER_ID, p.PROJECT_ID
FROM emp_record_table e
INNER JOIN proj_table p
WHERE e.ROLE NOT IN ("MANAGER","PRESIDENT","CEO")
ORDER BY e.MANAGER_ID;
```

Output:



The screenshot shows a database query editor with a toolbar and a query window. The query is as follows:

```
3 • SELECT e.EMP_ID , e.FIRST_NAME , e.LAST_NAME, e.DEPT, e.MANAGER_ID, p.PROJECT_ID
4 FROM emp_record_table e
5 INNER JOIN proj_table p
6 WHERE e.ROLE NOT IN ("MANAGER","PRESIDENT","CEO")
7 ORDER BY e.MANAGER_ID;
```

Below the query window, the 'Result Grid' is displayed, showing the results of the query. The grid has 7 columns: EMP_ID, FIRST_NAME, LAST_NAME, DEPT, MANAGER_ID, and PROJECT_ID. The results are as follows:

EMP_ID	FIRST_NAME	LAST_NAME	DEPT	MANAGER_ID	PROJECT_ID
E057	Dorothy	Wilson	HEALTHCARE	E083	P406
E052	Dianna	Wilson	HEALTHCARE	E083	P103
E057	Dorothy	Wilson	HEALTHCARE	E083	P103
E057	Dorothy	Wilson	HEALTHCARE	E083	P302
E057	Dorothy	Wilson	HEALTHCARE	E083	P204
E057	Dorothy	Wilson	HEALTHCARE	E083	P109
E052	Dianna	Wilson	HEALTHCARE	E083	P406
E505	Chad	Wilson	HEALTHCARE	E083	P109
E505	Chad	Wilson	HEALTHCARE	E083	P204
E057	Dorothy	Wilson	HEALTHCARE	E083	P105
E505	Chad	Wilson	HEALTHCARE	E083	P302
E505	Chad	Wilson	HEALTHCARE	E083	P406
E505	Chad	Wilson	HEALTHCARE	E083	P103
E052	Dianna	Wilson	HEALTHCARE	E083	P302
E052	Dianna	Wilson	HEALTHCARE	E083	P204
E052	Dianna	Wilson	HEALTHCARE	E083	P109
E052	Dianna	Wilson	HEALTHCARE	E083	P105
E057	Dorothy	Wilson	HEALTHCARE	E083	P105

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.

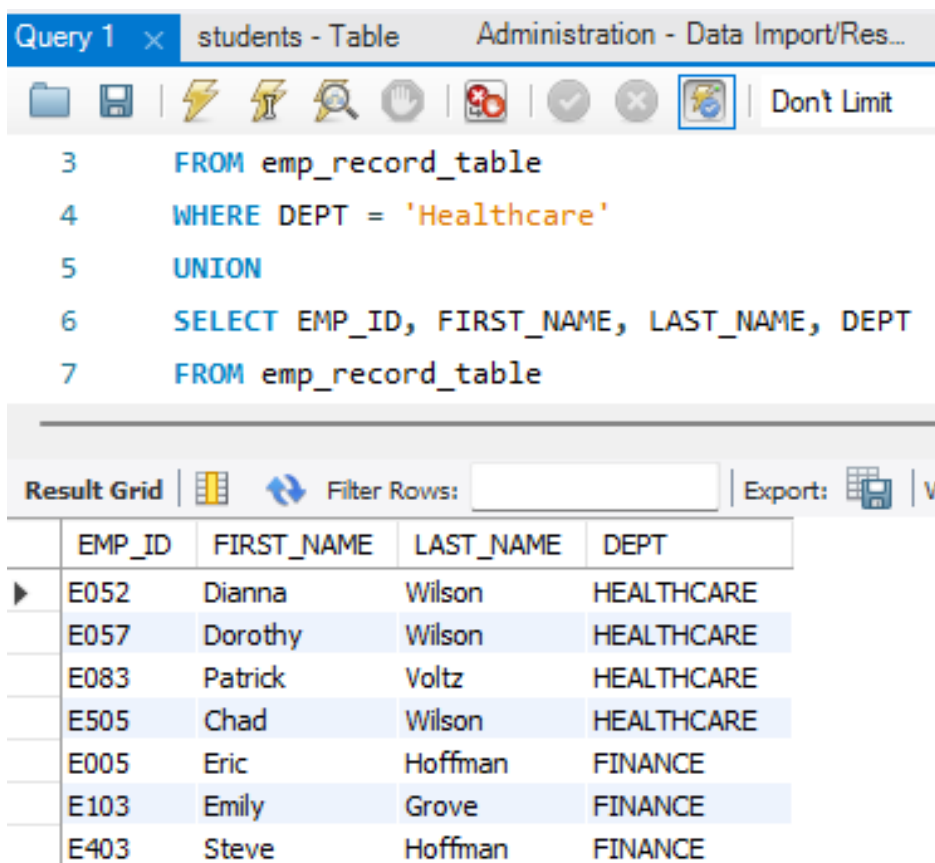
SQL code:

```
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';
```

Output:



The screenshot shows a SQL query editor window with the following query:

```
3 FROM emp_record_table
4 WHERE DEPT = 'Healthcare'
5 UNION
6 SELECT EMP_ID, FIRST_NAME, LAST_NAME, DEPT
7 FROM emp_record_table
```

Below the query editor is the 'Result Grid' showing the output of the query. The grid has 8 rows and 4 columns: EMP_ID, FIRST_NAME, LAST_NAME, and DEPT.

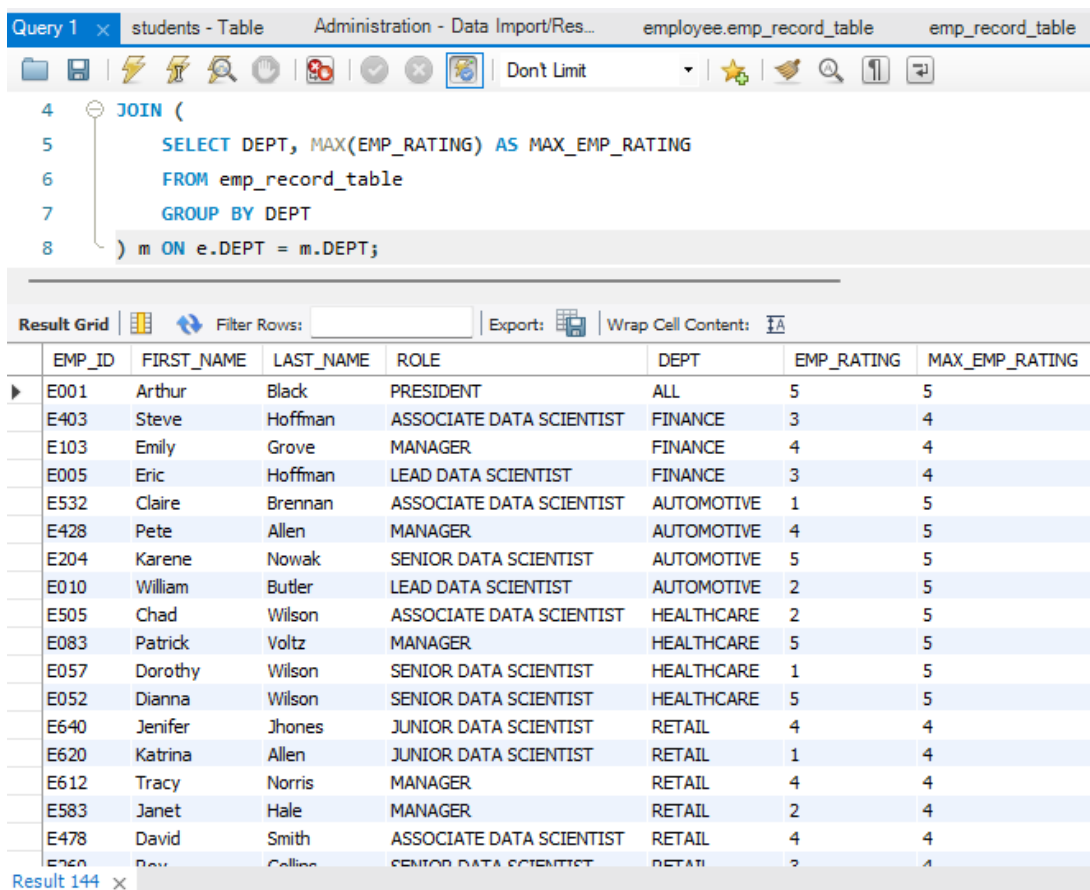
EMP_ID	FIRST_NAME	LAST_NAME	DEPT
E052	Dianna	Wilson	HEALTHCARE
E057	Dorothy	Wilson	HEALTHCARE
E083	Patrick	Voltz	HEALTHCARE
E505	Chad	Wilson	HEALTHCARE
E005	Eric	Hoffman	FINANCE
E103	Emily	Grove	FINANCE
E403	Steve	Hoffman	FINANCE

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.

SQL code:

```
SELECT e.EMP_ID, e.FIRST_NAME, e.LAST_NAME, e.ROLE, e.DEPT, e.EMP_RATING,
m.MAX_EMP_RATING
FROM emp_record_table e
JOIN (
    SELECT DEPT, MAX(EMP_RATING) AS MAX_EMP_RATING
    FROM emp_record_table
    GROUP BY DEPT
) m ON e.DEPT = m.DEPT;
```

Output:



The screenshot shows a SQL query editor with the following query:

```
4 JOIN (
5     SELECT DEPT, MAX(EMP_RATING) AS MAX_EMP_RATING
6     FROM emp_record_table
7     GROUP BY DEPT
8 ) m ON e.DEPT = m.DEPT;
```

Below the query, the 'Result Grid' is displayed, showing the output of the query. The grid has 8 columns: EMP_ID, FIRST_NAME, LAST_NAME, ROLE, DEPT, EMP_RATING, and MAX_EMP_RATING. The data is sorted by DEPT and then by EMP_RATING.

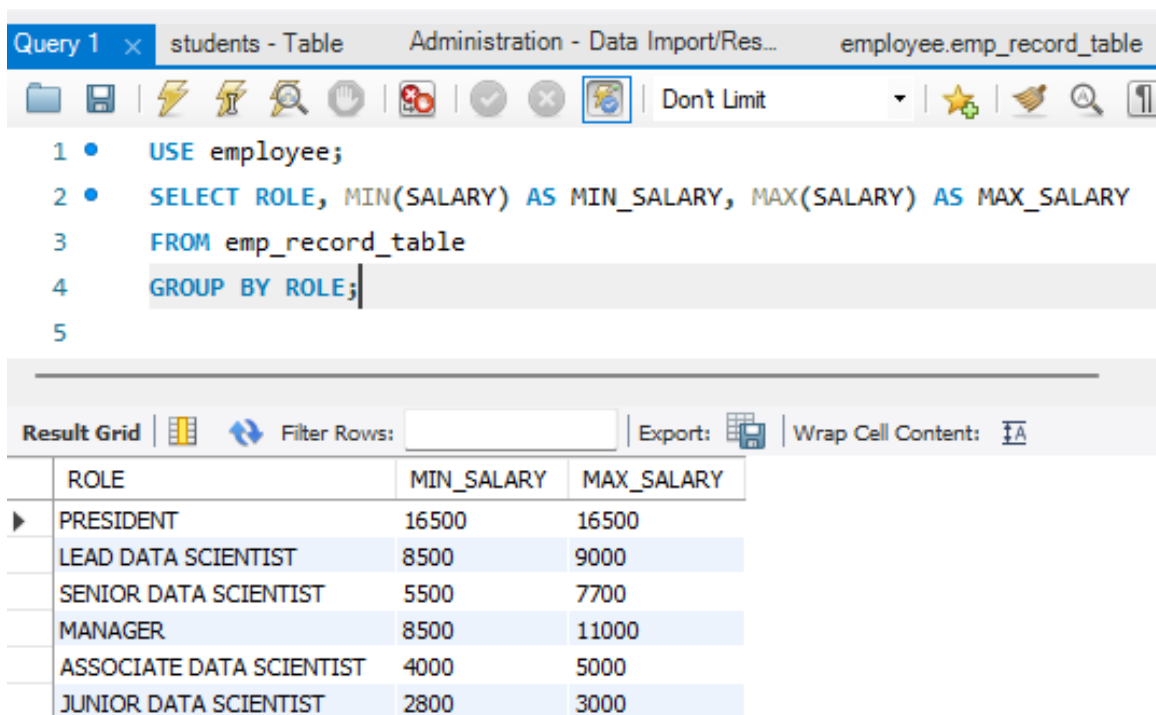
EMP_ID	FIRST_NAME	LAST_NAME	ROLE	DEPT	EMP_RATING	MAX_EMP_RATING
E001	Arthur	Black	PRESIDENT	ALL	5	5
E403	Steve	Hoffman	ASSOCIATE DATA SCIENTIST	FINANCE	3	4
E103	Emily	Grove	MANAGER	FINANCE	4	4
E005	Eric	Hoffman	LEAD DATA SCIENTIST	FINANCE	3	4
E532	Claire	Brennan	ASSOCIATE DATA SCIENTIST	AUTOMOTIVE	1	5
E428	Pete	Allen	MANAGER	AUTOMOTIVE	4	5
E204	Karene	Nowak	SENIOR DATA SCIENTIST	AUTOMOTIVE	5	5
E010	William	Butler	LEAD DATA SCIENTIST	AUTOMOTIVE	2	5
E505	Chad	Wilson	ASSOCIATE DATA SCIENTIST	HEALTHCARE	2	5
E083	Patrick	Voltz	MANAGER	HEALTHCARE	5	5
E057	Dorothy	Wilson	SENIOR DATA SCIENTIST	HEALTHCARE	1	5
E052	Dianna	Wilson	SENIOR DATA SCIENTIST	HEALTHCARE	5	5
E640	Jenifer	Jhones	JUNIOR DATA SCIENTIST	RETAIL	4	4
E620	Katrina	Allen	JUNIOR DATA SCIENTIST	RETAIL	1	4
E612	Tracy	Norris	MANAGER	RETAIL	4	4
E583	Janet	Hale	MANAGER	RETAIL	2	4
E478	David	Smith	ASSOCIATE DATA SCIENTIST	RETAIL	4	4
E760	David	Collier	SENIOR DATA SCIENTIST	RETAIL	3	4

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.

SQL code:

```
SELECT ROLE, MIN(SALARY) AS MIN_SALARY, MAX(SALARY) AS MAX_SALARY  
FROM emp_record_table  
GROUP BY ROLE;
```

Output:



The screenshot shows a database query editor with a toolbar and a result grid. The query editor has tabs for 'Query 1', 'students - Table', 'Administration - Data Import/Res...', and 'employee.emp_record_table'. The query text is as follows:

```
1 • USE employee;  
2 • SELECT ROLE, MIN(SALARY) AS MIN_SALARY, MAX(SALARY) AS MAX_SALARY  
3 FROM emp_record_table  
4 GROUP BY ROLE;  
5
```

The result grid below the query editor displays the output of the query. It has a toolbar with 'Result Grid', 'Filter Rows', 'Export', and 'Wrap Cell Content'. The data is as follows:

	ROLE	MIN_SALARY	MAX_SALARY
▶	PRESIDENT	16500	16500
	LEAD DATA SCIENTIST	8500	9000
	SENIOR DATA SCIENTIST	5500	7700
	MANAGER	8500	11000
	ASSOCIATE DATA SCIENTIST	4000	5000
	JUNIOR DATA SCIENTIST	2800	3000

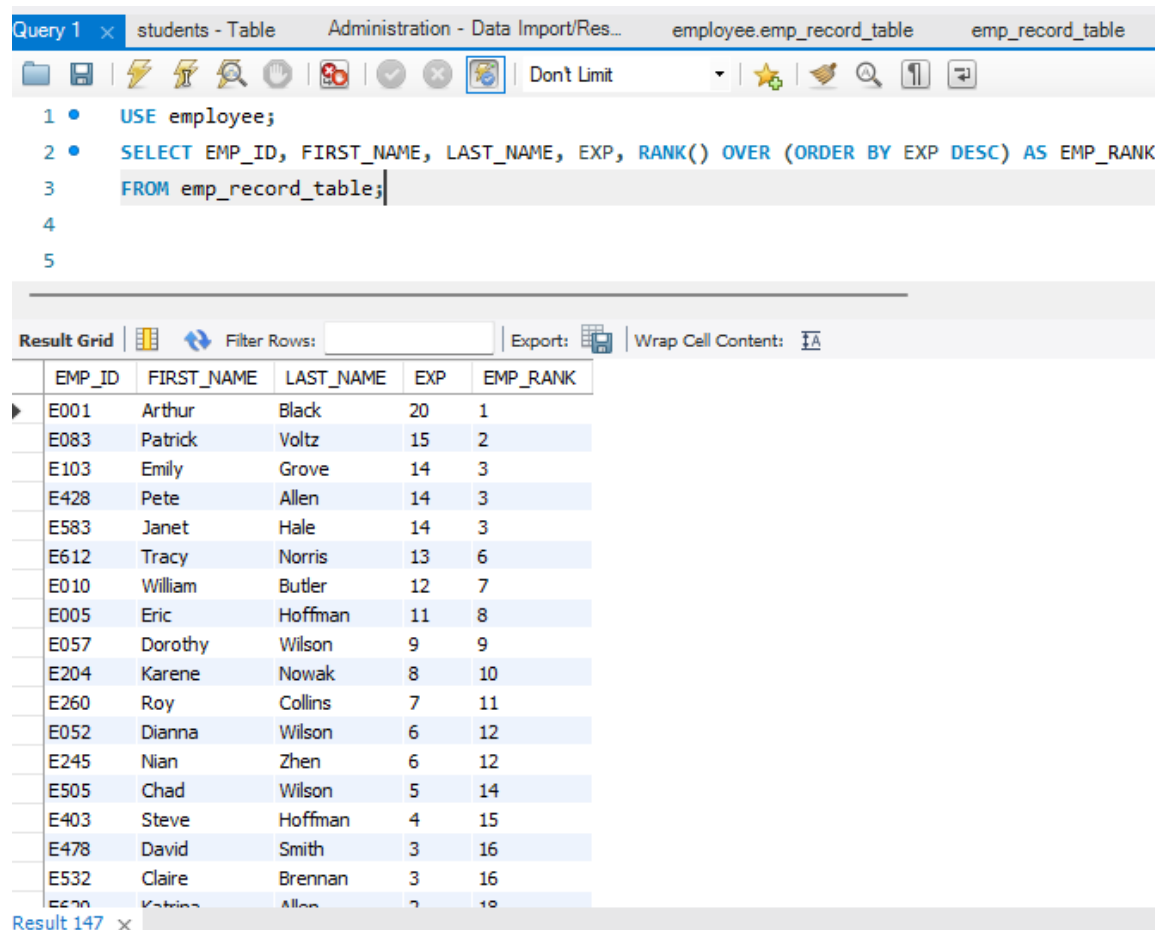
10. Write a query to assign ranks to each employee based on their experience. Take data from the employee record table.

SQL code:

```
SELECT EMP_ID, FIRST_NAME, LAST_NAME, EXP, RANK() OVER (ORDER BY EXP DESC) AS  
EMP_RANK
```

```
FROM emp_record_table;
```

Output:



The screenshot shows a database query editor with a query window and a result grid. The query window contains the following SQL code:

```
1 • USE employee;  
2 • SELECT EMP_ID, FIRST_NAME, LAST_NAME, EXP, RANK() OVER (ORDER BY EXP DESC) AS EMP_RANK  
3 • FROM emp_record_table;  
4  
5
```

The result grid displays the output of the query, showing a table with 5 columns: EMP_ID, FIRST_NAME, LAST_NAME, EXP, and EMP_RANK. The data is sorted by experience (EXP) in descending order. The first row is highlighted with a blue background.

EMP_ID	FIRST_NAME	LAST_NAME	EXP	EMP_RANK
E001	Arthur	Black	20	1
E083	Patrick	Voltz	15	2
E103	Emily	Grove	14	3
E428	Pete	Allen	14	3
E583	Janet	Hale	14	3
E612	Tracy	Norris	13	6
E010	William	Butler	12	7
E005	Eric	Hoffman	11	8
E057	Dorothy	Wilson	9	9
E204	Karene	Nowak	8	10
E260	Roy	Collins	7	11
E052	Dianna	Wilson	6	12
E245	Nian	Zhen	6	12
E505	Chad	Wilson	5	14
E403	Steve	Hoffman	4	15
E478	David	Smith	3	16
E532	Claire	Brennan	3	16
E600	Katrina	Allen	2	18

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.

SQL code:

```
CREATE VIEW high_salary_employees_view AS  
SELECT EMP_ID, FIRST_NAME, LAST_NAME, COUNTRY, SALARY  
FROM emp_record_table  
WHERE SALARY > 6000;
```

Output:

The screenshot shows a database management tool interface. At the top, there are tabs for 'Query 1', 'students - Table', 'Administration - Data Import/Res...', 'employee.emp_record_table', and 'emp_record_table'. Below the tabs is a toolbar with various icons. The main area displays the following SQL code:

```
4  
5 • CREATE VIEW high_salary_employees_view AS  
6 SELECT EMP_ID, FIRST_NAME, LAST_NAME, COUNTRY, SALARY  
7 FROM emp_record_table  
8 WHERE SALARY > 6000;
```

Below the code editor is a 'Result Grid' section with a table showing the details of the created view:

View	Create View	character_set_client	collation_connection
high_salary_employees_view	CREATE ALGORITHM=UNDEFINED DEFINER='r...' utf8mb4	utf8mb4	utf8mb4_0900_ai_ci

At the bottom, there is a 'Result 148' section with an 'Output' tab. The 'Action Output' tab is selected, showing a table with the following data:

#	Time	Action	Message
✓ 1	20:36:33	CREATE VIEW high_salary_employees_view AS SELECT EMP_ID, FIRST_NAME, L...	0 row(s) affected
✓ 2	20:38:11	SHOW CREATE VIEW high_salary_employees_view	1 row(s) returned

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

SQL code:

```
SELECT *  
FROM emp_record_table  
WHERE EMP_ID IN (  
    SELECT EMP_ID  
    FROM emp_record_table  
    WHERE EXP > 10  
);
```

Output:

The screenshot shows a database query editor with a query window and a results window. The query window contains a nested query to find employees with more than 10 years of experience. The results window displays a table of employee records and an action log.

Query 1:

```
5 WHERE EMP_ID IN (  
6     SELECT EMP_ID  
7     FROM emp_record_table  
8     WHERE EXP > 10  
9 )
```

Result Grid:

	EMP_ID	FIRST_NAME	LAST_NAME	GENDER	ROLE	DEPT	EXP	COUNTRY	CONTINENT	SALARY	EMP_RATING	MANAGER_ID	PROJ_ID
▶	E001	Arthur	Black	M	PRESIDENT	ALL	20	USA	NORTH AMERICA	16500	5	NULL	NULL
	E005	Eric	Hoffman	M	LEAD DATA SCIENTIST	FINANCE	11	USA	NORTH AMERICA	8500	3	E103	P105
	E010	William	Butler	M	LEAD DATA SCIENTIST	AUTOMOTIVE	12	FRANCE	EUROPE	9000	2	E428	P204
	E083	Patrick	Voltz	M	MANAGER	HEALTHCARE	15	USA	NORTH AMERICA	9500	5	E001	NULL
	E103	Emily	Grove	F	MANAGER	FINANCE	14	CANADA	NORTH AMERICA	10500	4	E001	NULL
	E428	Pete	Allen	M	MANAGER	AUTOMOTIVE	14	GERMANY	EUROPE	11000	4	E001	NULL
	E583	Janet	Hale	F	MANAGER	RETAIL	14	COLOMBIA	SOUTH AMERICA	10000	2	E001	NULL
	E612	Tracy	Norris	F	MANAGER	RETAIL	13	INDIA	ASIA	8500	4	E001	NULL

emp_record_table 149

Output

Action Output

#	Time	Action	Message
✓ 1	20:36:33	CREATE VIEW high_salary_employees_view AS SELECT EMP_ID, FIRST_NAME, LAST_NAME, COUNT...	0 row(s) affected
✓ 2	20:38:11	SHOW CREATE VIEW high_salary_employees_view	1 row(s) returned
✓ 3	20:39:57	SELECT * FROM emp_record_table WHERE EMP_ID IN (SELECT EMP_ID FROM emp_record_tabl...	8 row(s) returned

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.

SQL code:

DELIMITER &&

CREATE PROCEDURE get_experience()

BEGIN

SELECT * FROM emp_record_table WHERE EXP > 3;

END &&

DELIMITER ;

Output:

The screenshot displays a database management interface with a query editor and a results pane. The query editor shows the following SQL code:

```
10  
11 END &&  
12 DELIMITER ;  
13  
14 CALL get_experience();
```

The results pane shows a table with 14 columns: EMP_ID, FIRST_NAME, LAST_NAME, GENDER, ROLE, DEPT, EXP, COUNTRY, CONTINENT, SALARY, EMP_RATING, MANAGER_ID, and PROJ_ID. The table contains 15 rows of data, including employees like Arthur Black (President), Eric Hoffman (Lead Data Scientist), and William Butler (Lead Data Scientist).

Below the table, the 'Output' pane shows the execution log:

#	Time	Action	Message
4	20:42:15	USE employee	0 row(s) affected
5	20:42:19	CREATE PROCEDURE get_experience() BEGIN SELECT * FROM emp_record_table WHERE EXP > 3...	0 row(s) affected
6	20:42:32	CALL get_experience()	15 row(s) returned

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'.

SQL code:

```
SELECT
    dt.EMP_ID,
    dt.FIRST_NAME,
    dt.LAST_NAME,
    dt.EXP,
    CASE
        WHEN dt.EXP <= 2 THEN 'JUNIOR DATA SCIENTIST'
        WHEN dt.EXP > 2 AND dt.EXP <= 5 THEN 'ASSOCIATE DATA SCIENTIST'
        WHEN dt.EXP > 5 AND dt.EXP <= 10 THEN 'SENIOR DATA SCIENTIST'
        WHEN dt.EXP > 10 AND dt.EXP <= 12 THEN 'LEAD DATA SCIENTIST'
        WHEN dt.EXP > 12 AND dt.EXP <= 16 THEN 'MANAGER'
    END AS JOB_PROFILE
FROM
    data_science_team dt
LEFT JOIN
    (
```

```

SELECT

    PROJECT_ID,

    ROW_NUMBER() OVER (PARTITION BY PROJECT_ID ORDER BY START_DATE
DESC) AS rn

FROM

    proj_table

) pt ON dt.EMP_ID = pt.PROJECT_ID AND pt.rn = 1;

```

Output:

Query 1 x students - Table Administration - Data Import/Res... employee.emp_record_table emp_record_table

1 • USE employee;

2

3 • SELECT

4 dt.EMP_ID,

5 dt.FIRST_NAME,

Result Grid | Filter Rows: | Export: | Wrap Cell Contents: |

	EMP_ID	FIRST_NAME	LAST_NAME	EXP	JOB_PROFILE
▶	E005	Eric	Hoffman	11	LEAD DATA SCIENTIST
	E010	William	Butler	12	LEAD DATA SCIENTIST
	E052	Dianna	Wilson	6	SENIOR DATA SCIENTIST
	E057	Dorothy	Wilson	9	SENIOR DATA SCIENTIST
	E204	Karene	Nowak	8	SENIOR DATA SCIENTIST
	E245	Nian	Zhen	6	SENIOR DATA SCIENTIST
	E260	Roy	Collins	7	SENIOR DATA SCIENTIST
	E403	Steve	Hoffman	4	ASSOCIATE DATA SCIENTIST
	E478	David	Smith	3	ASSOCIATE DATA SCIENTIST

Result 151 x

Output

Action Output

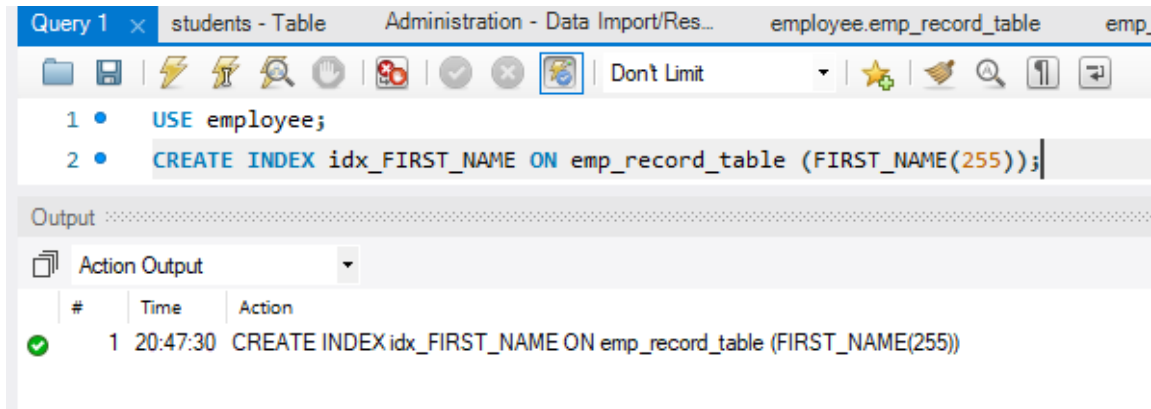
#	Time	Action	Message
✓ 5	20:42:19	CREATE PROCEDURE get_experience() BEGIN SELECT * FROM emp_record_table WHERE EXP > 3...	0 row(s) affected
✓ 6	20:42:32	CALL get_experience()	15 row(s) returned
✓ 7	20:44:09	SELECT dt.EMP_ID, dt.FIRST_NAME, dt.LAST_NAME, dt.EXP, CASE WHEN dt.EXP ...	13 row(s) returned

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.

SQL code:

```
CREATE INDEX idx_FIRST_NAME ON emp_record_table (FIRST_NAME(255));
```

Output:



The screenshot shows a database management tool interface. At the top, there are tabs for 'Query 1', 'students - Table', 'Administration - Data Import/Res...', 'employee.emp_record_table', and 'emp.'. Below the tabs is a toolbar with various icons. The main area displays two SQL statements:

- 1 • `USE employee;`
- 2 • `CREATE INDEX idx_FIRST_NAME ON emp_record_table (FIRST_NAME(255));`

Below the SQL statements is an 'Output' section. It contains a table with the following data:

#	Time	Action
1	20:47:30	CREATE INDEX idx_FIRST_NAME ON emp_record_table (FIRST_NAME(255))

A green checkmark icon is visible next to the first row of the output table.

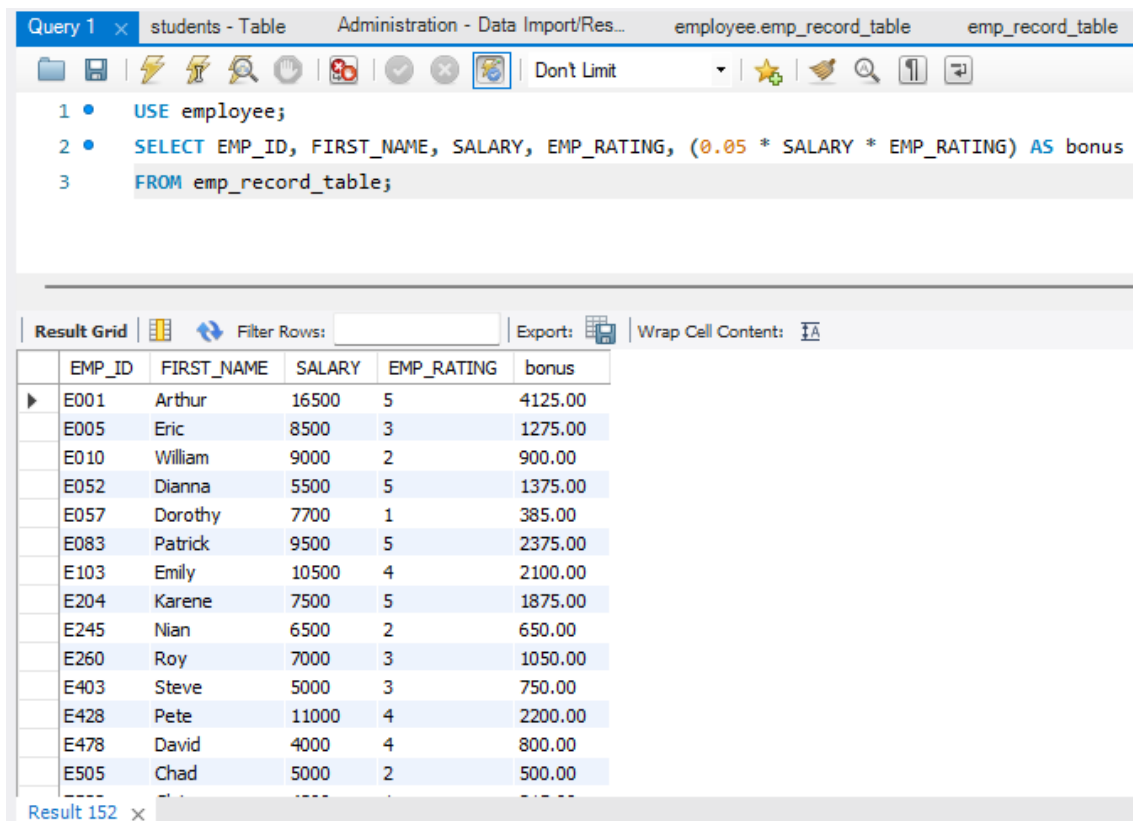
16. Write a query to calculate the bonus for all the employees, based on their ratings and salaries (Use the formula: 5% of salary * employee rating).

SQL code:

```
SELECT EMP_ID, FIRST_NAME, SALARY, EMP_RATING, (0.05 * SALARY * EMP_RATING) AS bonus
```

```
FROM emp_record_table;
```

Output:



The screenshot shows a database query editor with a toolbar at the top. The query is as follows:

```
1 • USE employee;  
2 • SELECT EMP_ID, FIRST_NAME, SALARY, EMP_RATING, (0.05 * SALARY * EMP_RATING) AS bonus  
3 • FROM emp_record_table;
```

Below the query editor is the 'Result Grid' section. It includes a 'Filter Rows' input field, an 'Export' button, and a 'Wrap Cell Content' checkbox. The results are displayed in a table with the following columns: EMP_ID, FIRST_NAME, SALARY, EMP_RATING, and bonus. The table contains 15 rows of data.

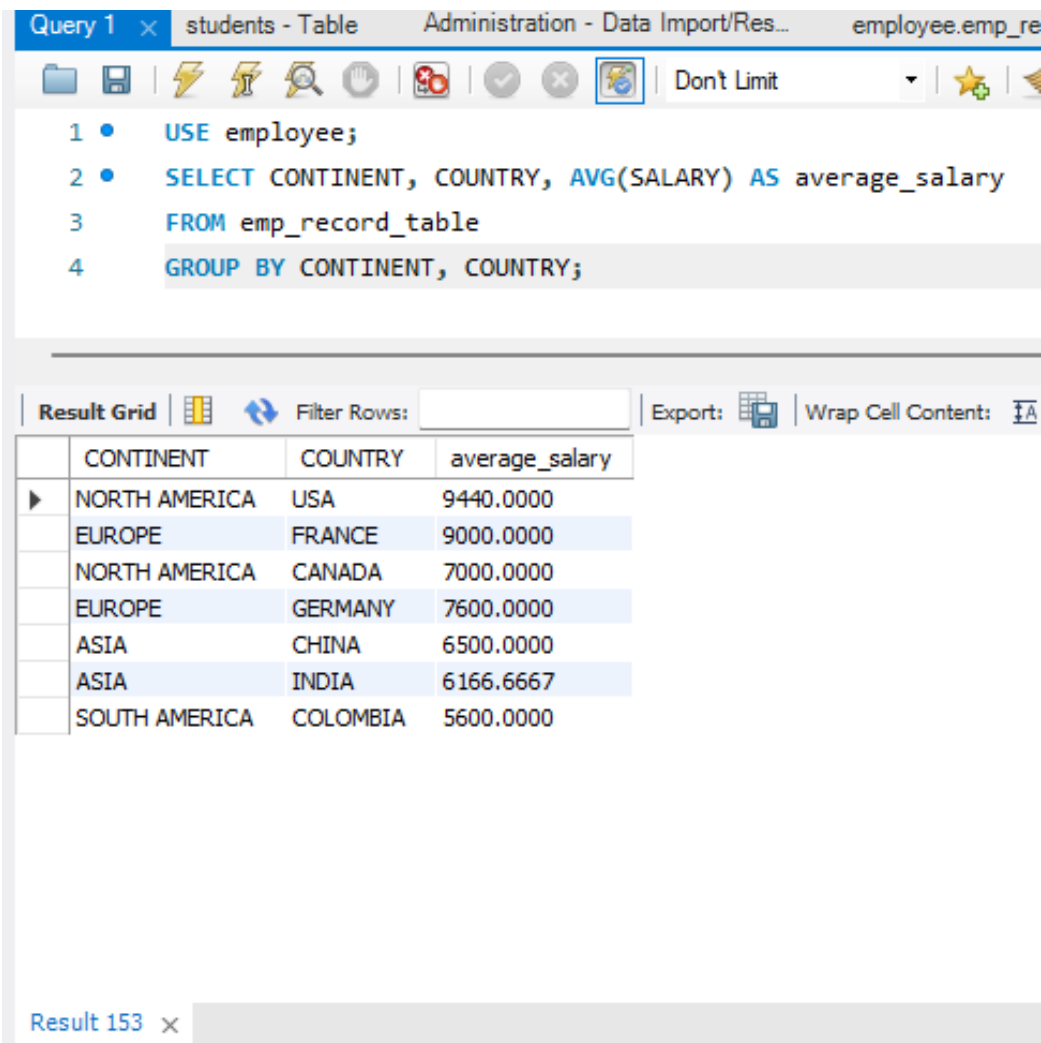
EMP_ID	FIRST_NAME	SALARY	EMP_RATING	bonus
E001	Arthur	16500	5	4125.00
E005	Eric	8500	3	1275.00
E010	William	9000	2	900.00
E052	Dianna	5500	5	1375.00
E057	Dorothy	7700	1	385.00
E083	Patrick	9500	5	2375.00
E103	Emily	10500	4	2100.00
E204	Karene	7500	5	1875.00
E245	Nian	6500	2	650.00
E260	Roy	7000	3	1050.00
E403	Steve	5000	3	750.00
E428	Pete	11000	4	2200.00
E478	David	4000	4	800.00
E505	Chad	5000	2	500.00

17. 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 emp_record_table  
GROUP BY CONTINENT, COUNTRY;
```

Output:



The screenshot shows a database query editor with a toolbar at the top. The query is as follows:

```
1 • USE employee;  
2 • SELECT CONTINENT, COUNTRY, AVG(SALARY) AS average_salary  
3   FROM emp_record_table  
4   GROUP BY CONTINENT, COUNTRY;
```

Below the query editor, the results are displayed in a grid format. The grid has four columns: CONTINENT, COUNTRY, and average_salary. The results are as follows:

CONTINENT	COUNTRY	average_salary
NORTH AMERICA	USA	9440.0000
EUROPE	FRANCE	9000.0000
NORTH AMERICA	CANADA	7000.0000
EUROPE	GERMANY	7600.0000
ASIA	CHINA	6500.0000
ASIA	INDIA	6166.6667
SOUTH AMERICA	COLOMBIA	5600.0000

At the bottom of the screenshot, there is a tab labeled "Result 153" with a close button (X).