

Scienceqtech Employee Performance Mapping

SQL PROJECT

VANDANA T V | IITK DATASCIENCE | 30/07/2024

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.

Create a database ‘employee’ using the query:

```
CREATE DATABASE employee;
```

Output:

The screenshot shows the MySQL Workbench interface. In the top-left pane, under 'SCHEMAS', the 'employee' database is selected, indicated by a red arrow. The main pane contains a SQL query window titled 'Query 1' with the following code:

```
-- Creating the database named employee
CREATE DATABASE employee;
```

Below the query window, the 'Information' tab is selected in the bottom-left panel, which displays the message 'No object selected'. At the bottom of the screen, the 'Object Info' and 'Session' tabs are visible. The bottom-right pane shows the execution log with the following entry:

#	Time	Action	Message	Duration / Fetch
1	20-20-52	CREATE DATABASE employee	1 row(s) affected	0.000 sec

To select the database ‘employee’ as a default database schema and perform the operations in it we have the query:

```
USE employee;
```

Output:

The screenshot shows the MySQL Workbench interface. In the Navigator pane, under the SCHEMAS section, the 'employee' database is selected and highlighted with a red box. The Query Editor pane contains the following SQL code:

```
1 -- Creating the database named employee
2 • CREATE DATABASE employee;
3
4 -- To set the database employee as the default schema
5 • USE employee;
```

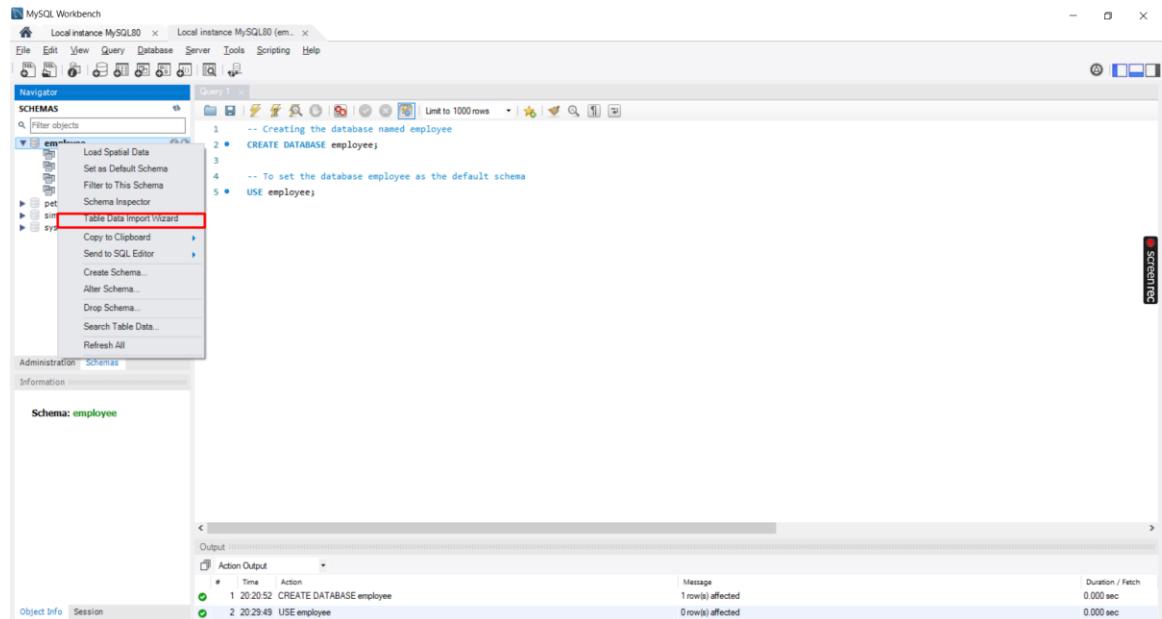
In the Output pane, the results of the executed queries are shown:

Action	Time	Action	Message	Duration / Fetch
1	20:20:52	CREATE DATABASE employee	1 row(s) affected	0.000 sec
2	20:29:49	USE employee	0 row(s) affected	0.000 sec

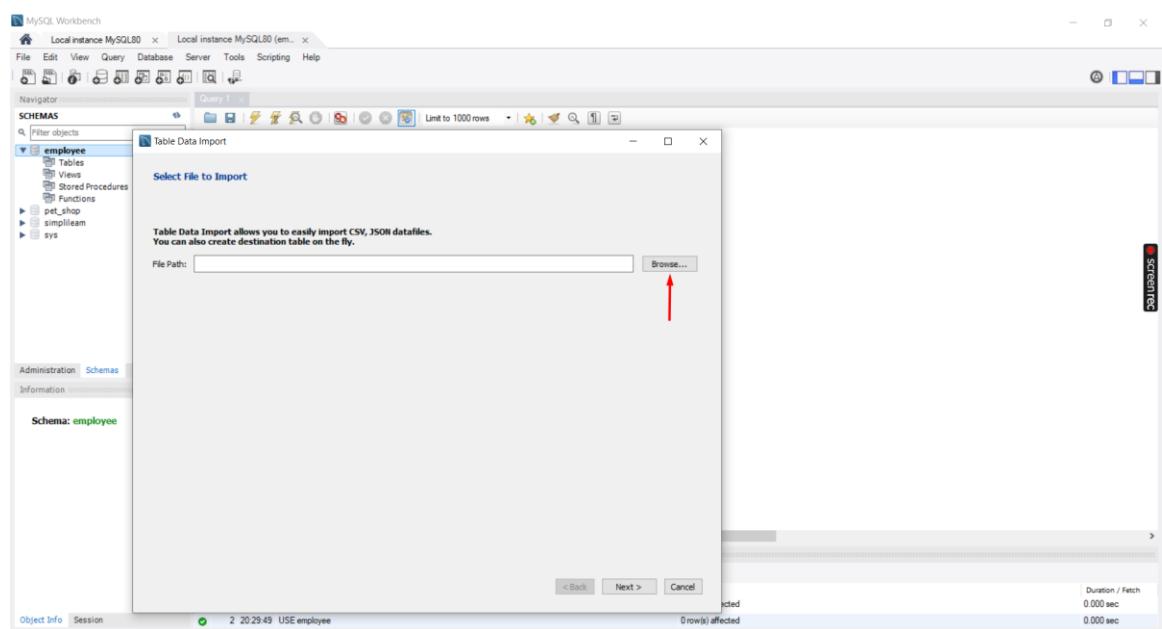
Importing Tables to the database:

Following are the steps to import the table data_science_team to the database employee

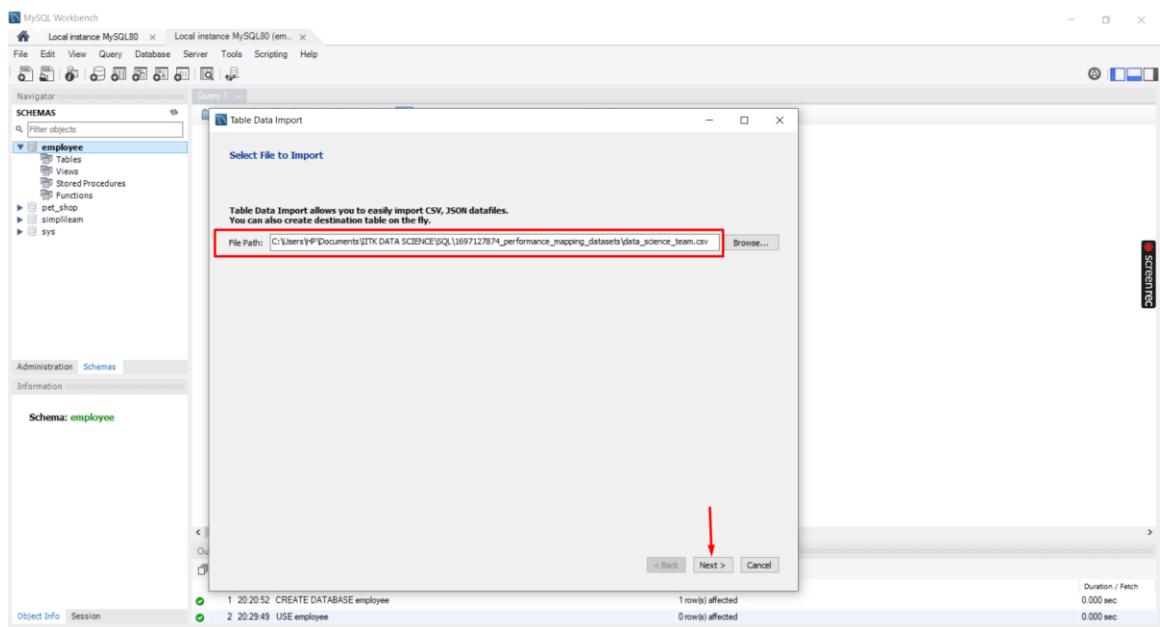
Step 1 : Right click on the employee database and choose ‘Table Data Import Wizard’ as shown below:



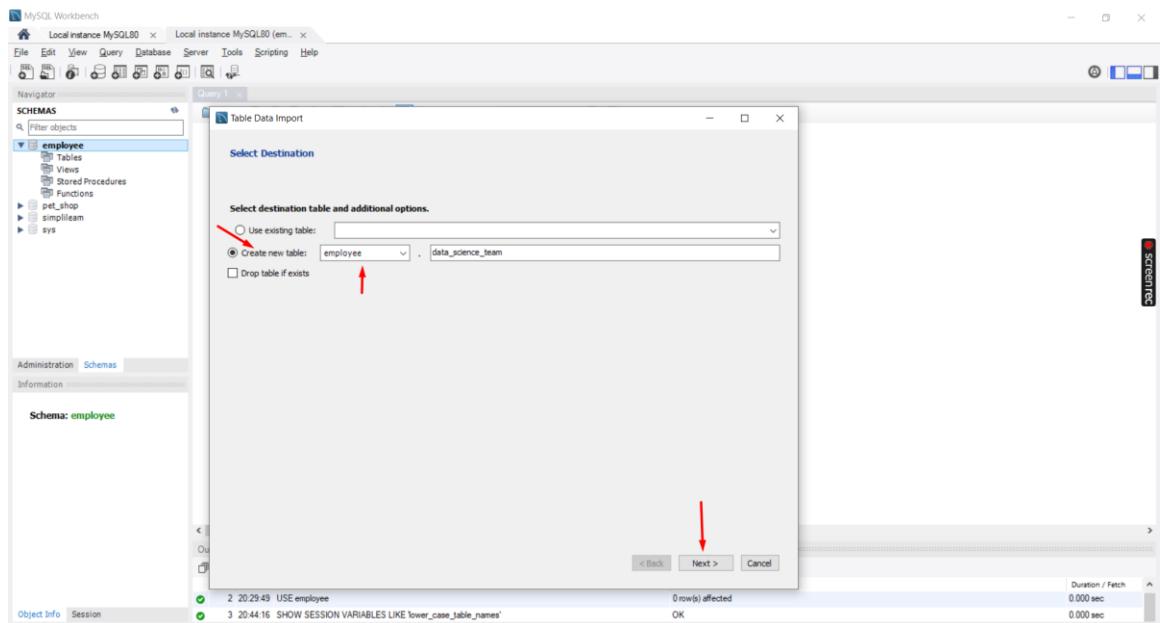
Step 2: Click on ‘Browse’ to select the file data_science_team.csv from the right location where the .csv file is saved.



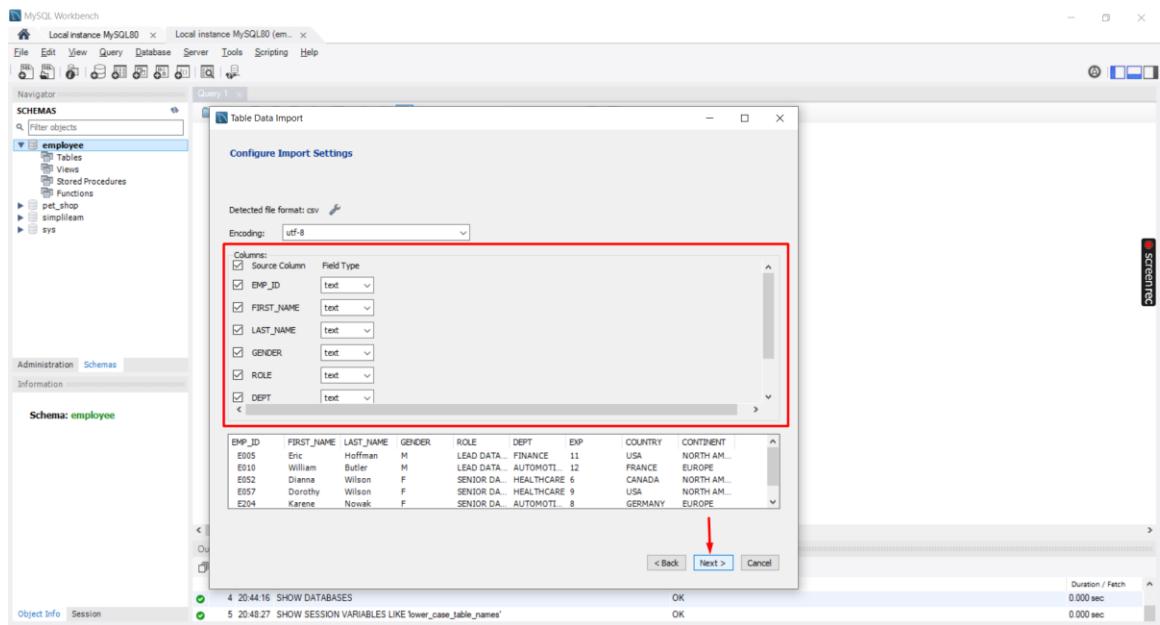
Step 3: Select the file data_science_team.csv from the saved location and click Next.



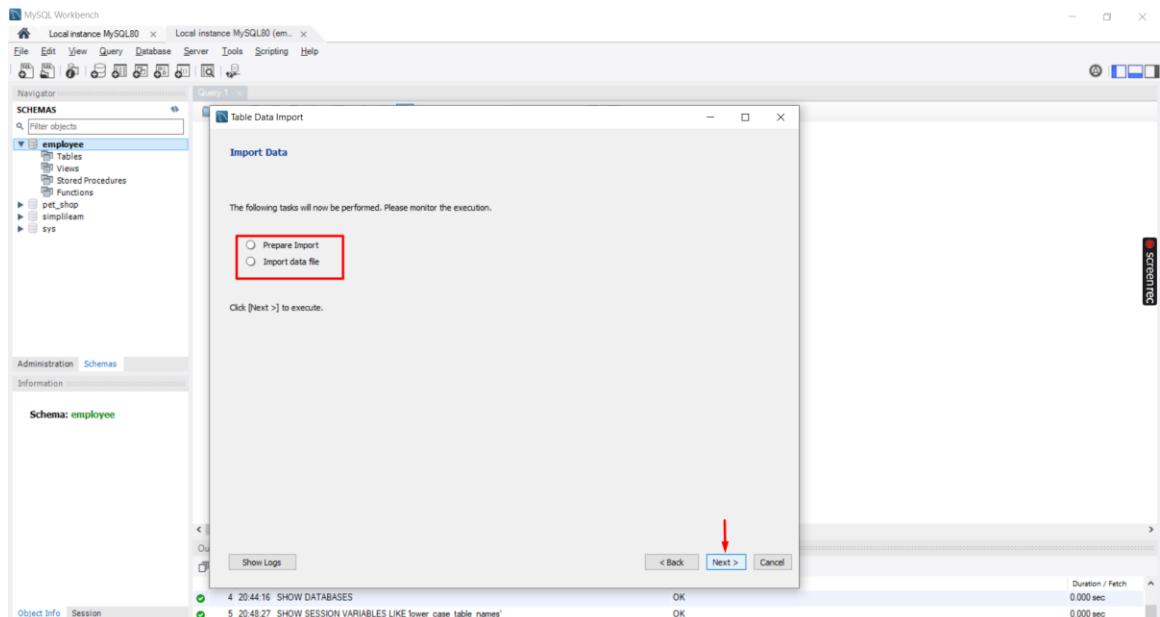
Step 4: Select ‘Create new table’ and make sure that the database is ‘employee’ then click Next.



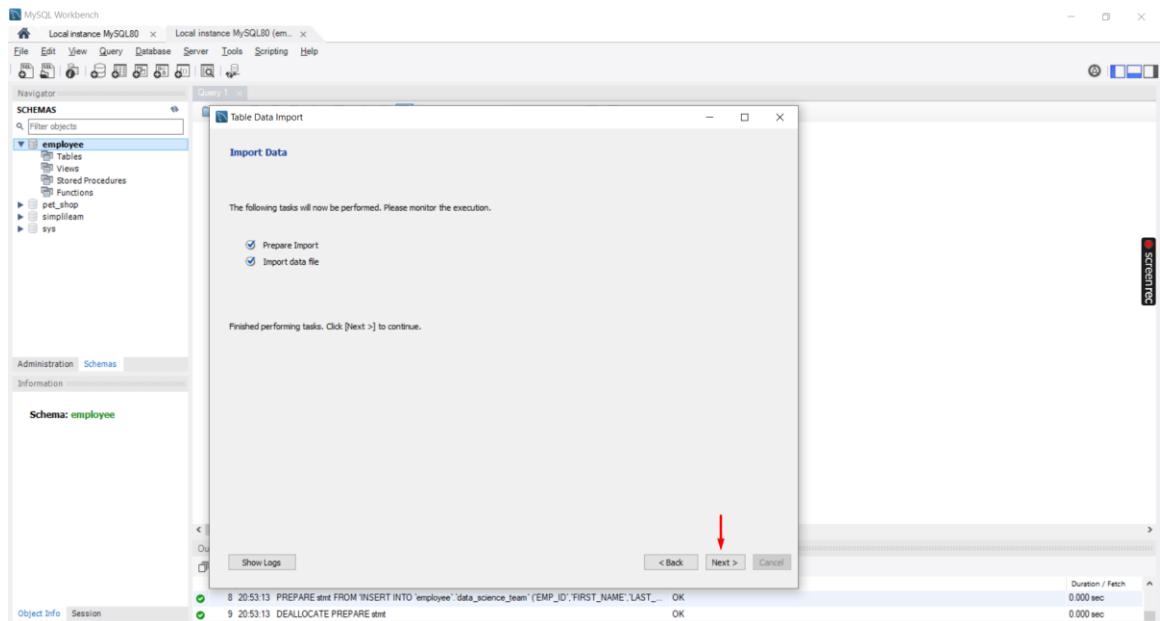
Step 5: Verify all the datatypes from the table which will be imported and click on Next.



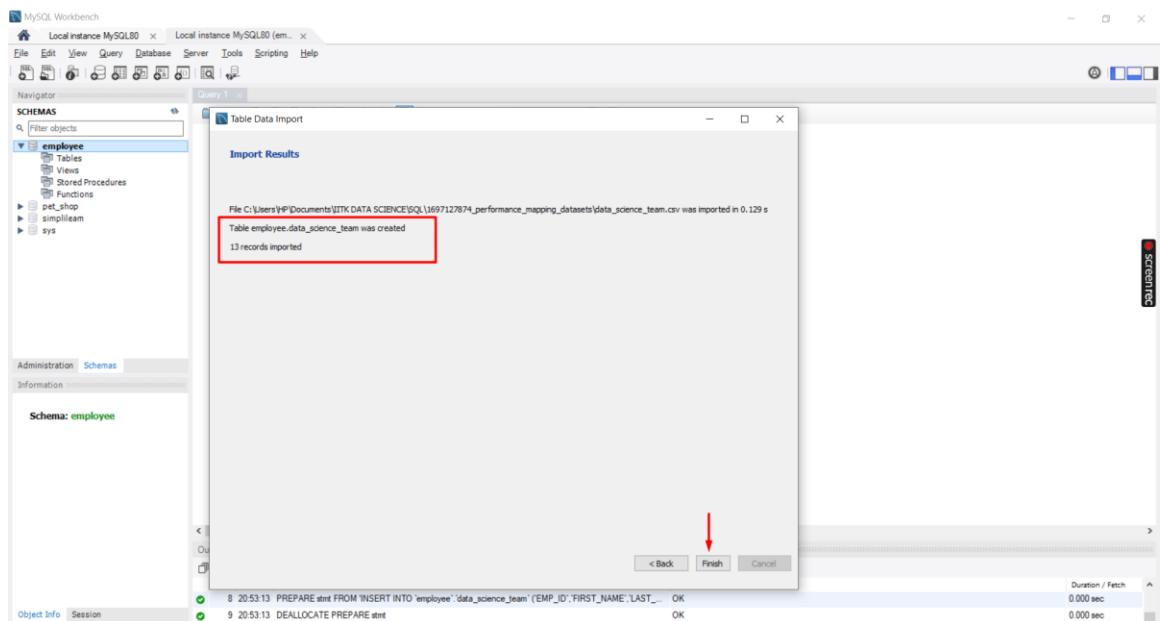
Step 6: In this window, we can see the steps which will be performed while importing the table and click Next which will start importing the table.



Step 7: Click Next in this window.



Step 8: Here in the below pop up window we can see the table has been created with 13 records. Click Finish to complete the creation of table named data_science_team.



After finishing all the steps we can see the table ‘data_science_team’ in the database schema ‘employee’.

```

-- Creating the database named employee
CREATE DATABASE employee;
-- To set the database employee as the default schema
USE employee;

```

#	Time	Action	Message	Duration / Fetch
8	20:53:13	PREPARE stmt FROM INSERT INTO `employee`.`data_science_team`(`EMP_ID`,`FIRST_NAME`,`LAST_NAME`,`GENDER`,`ROLE`,`DEPT`,`EXP`,`COUNTRY`,`CONTINENT`)	OK	0.000 sec
9	20:53:13	DEALLOCATE PREPARE stmt	OK	0.000 sec

Following are the steps to import the table ‘emp_record_table’ to the database ‘employee’.

Step 1 : Right click on the employee database and choose ‘Table Data Import Wizard’ as shown below

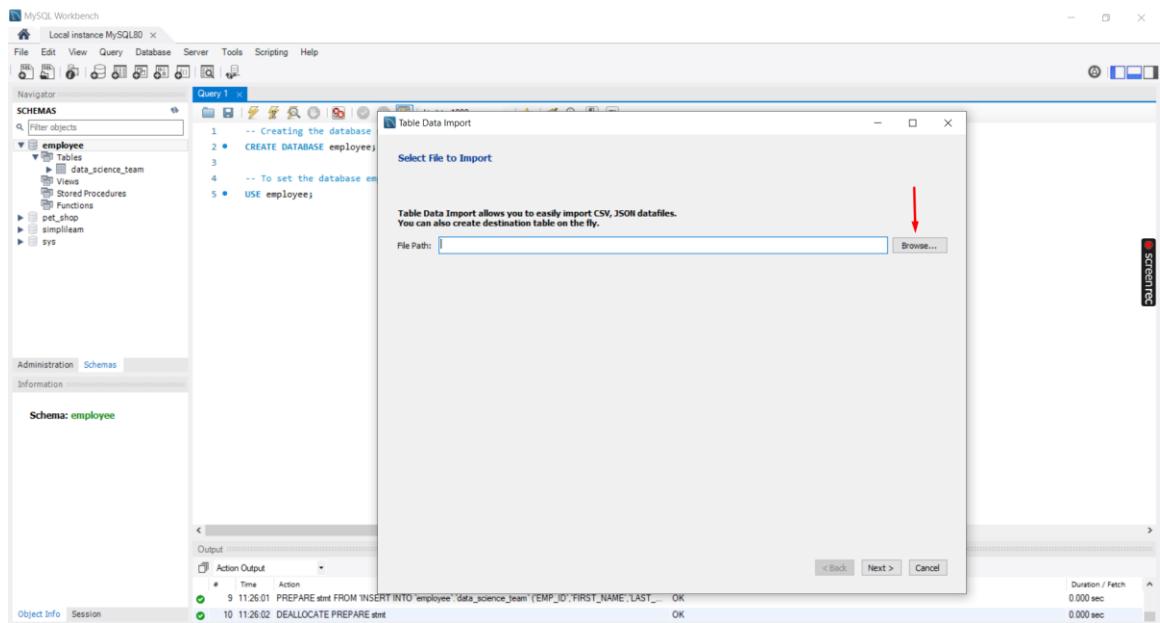
```

-- Creating the database named employee
CREATE DATABASE employee;
-- To set the database employee as the default schema
USE employee;

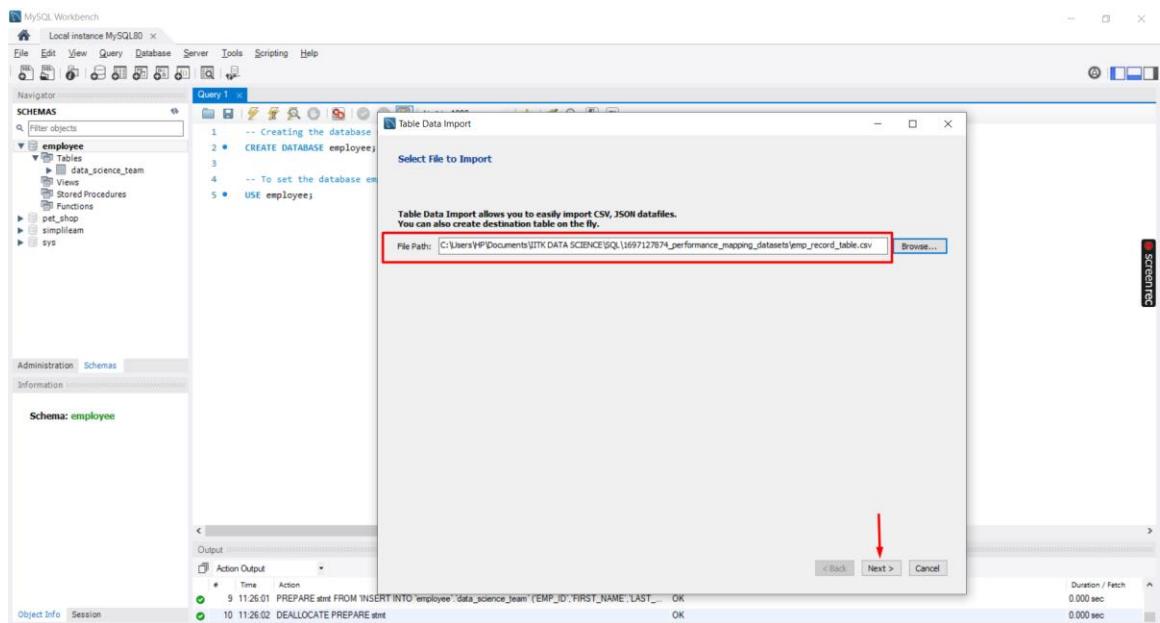
```

#	Time	Action	Message	Duration / Fetch
9	11:26:01	PREPARE stmt FROM INSERT INTO `employee`.`data_science_team`(`EMP_ID`,`FIRST_NAME`,`LAST_NAME`,`GENDER`,`ROLE`,`DEPT`,`EXP`,`COUNTRY`,`CONTINENT`)	OK	0.000 sec
10	11:26:02	DEALLOCATE PREPARE stmt	OK	0.000 sec

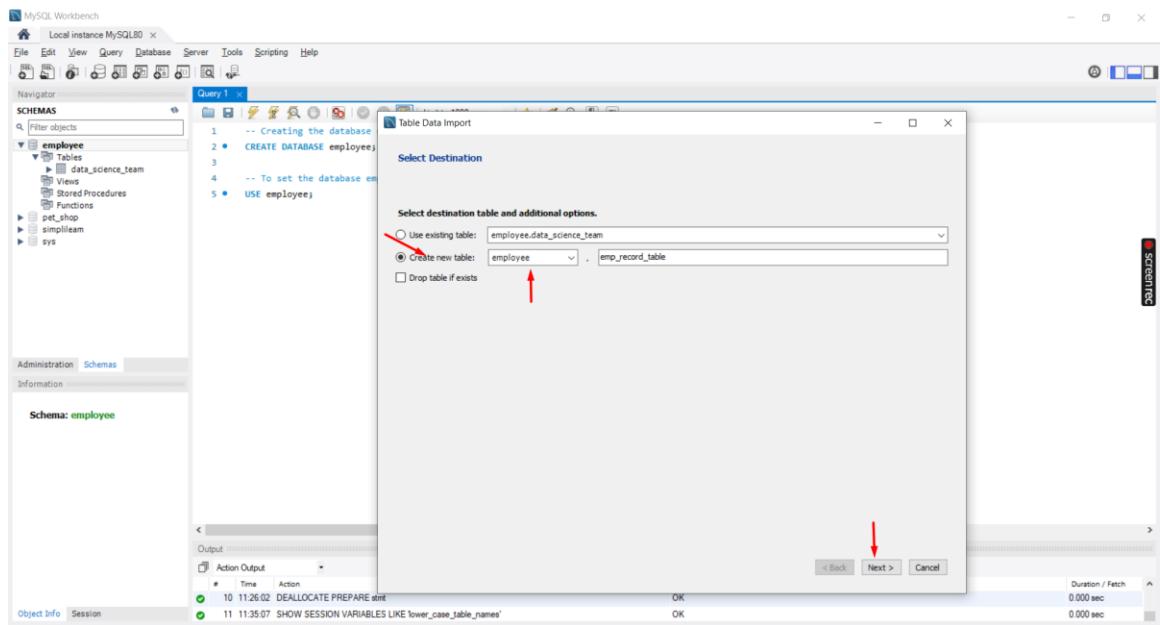
Step 2: Click on ‘Browse’ to select the file emp_record_table.csv from the right location where the .csv file is saved.



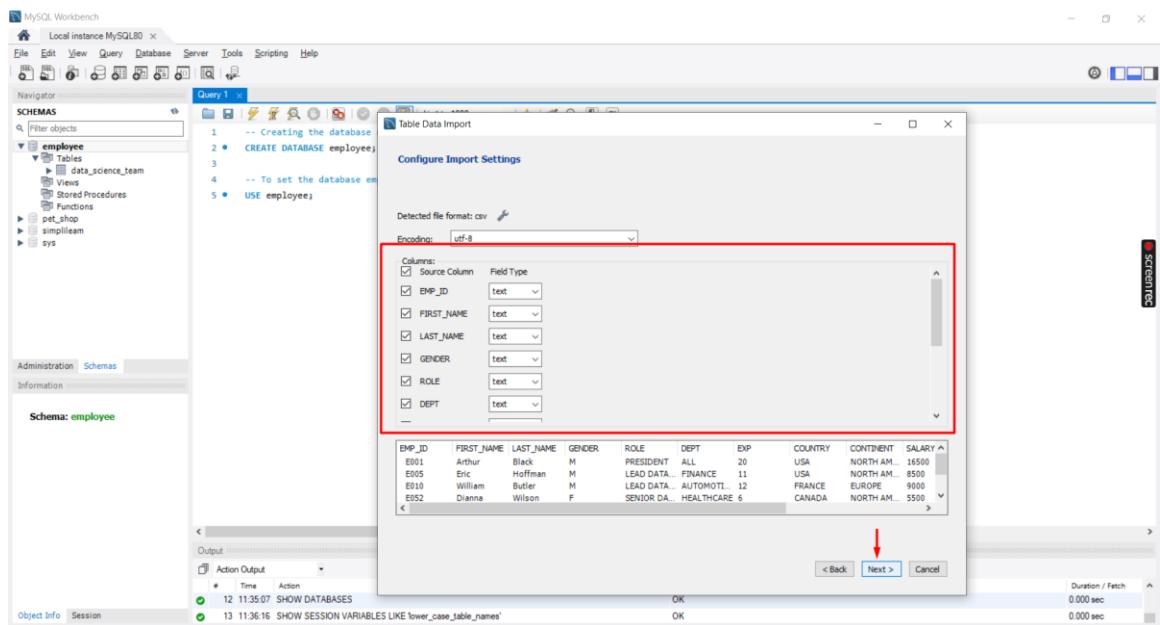
Step 3: Select the file emp_record_table.csv from the saved location and click Next.



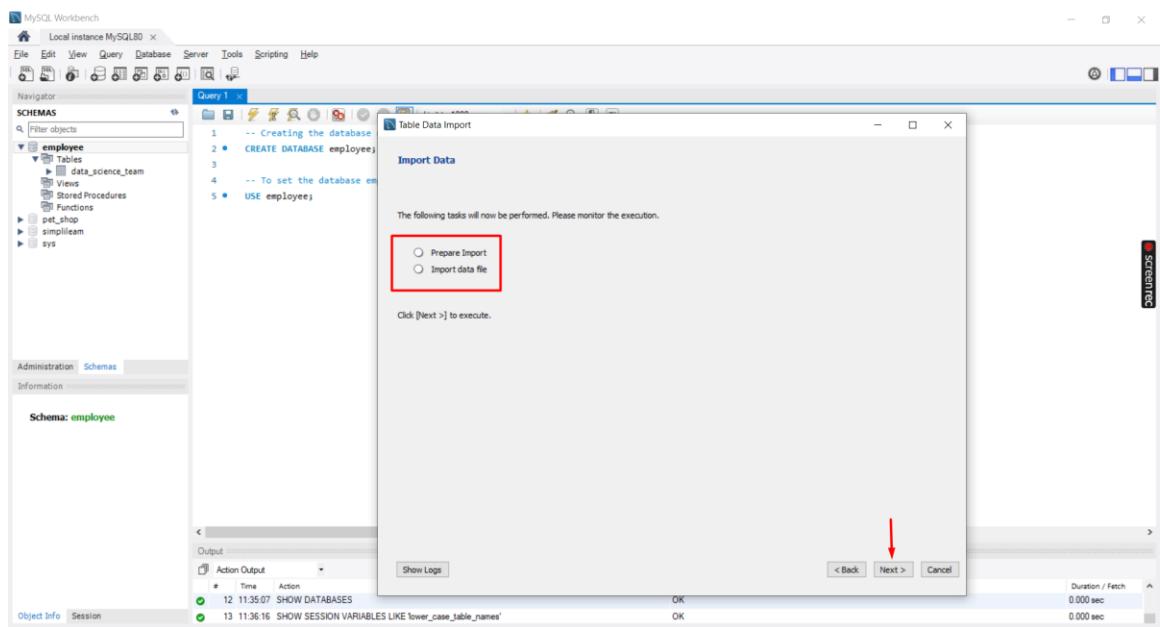
Step 4: Select ‘Create new table’ and make sure that the database is ‘employee’ then click Next.



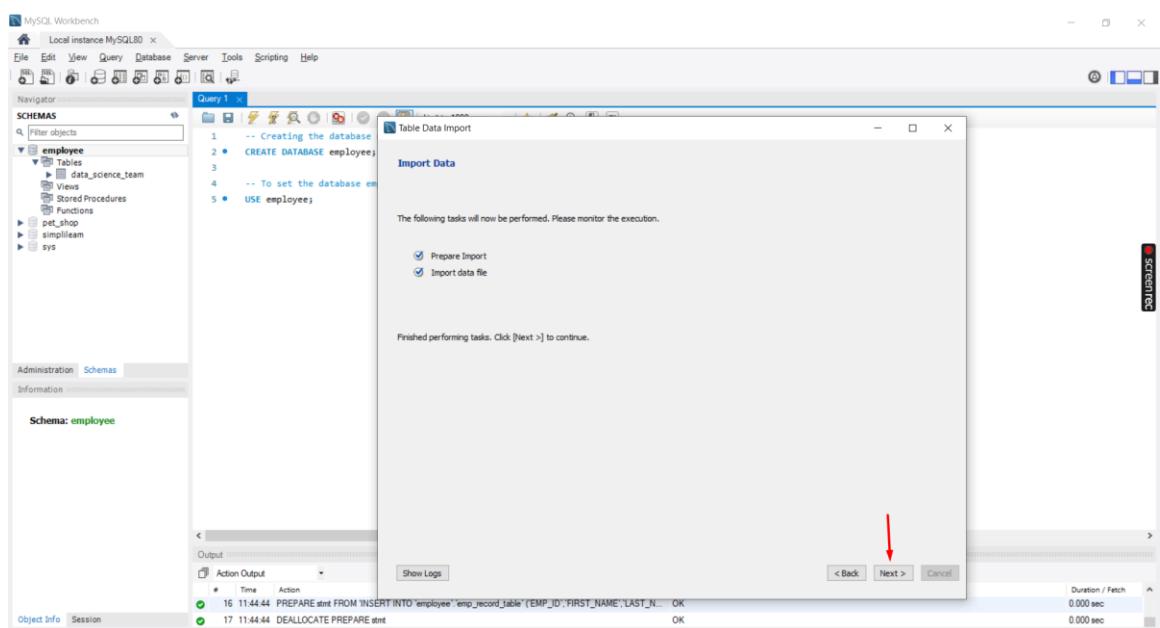
Step 5: Verify all the datatypes from the table which will be imported and click on Next.



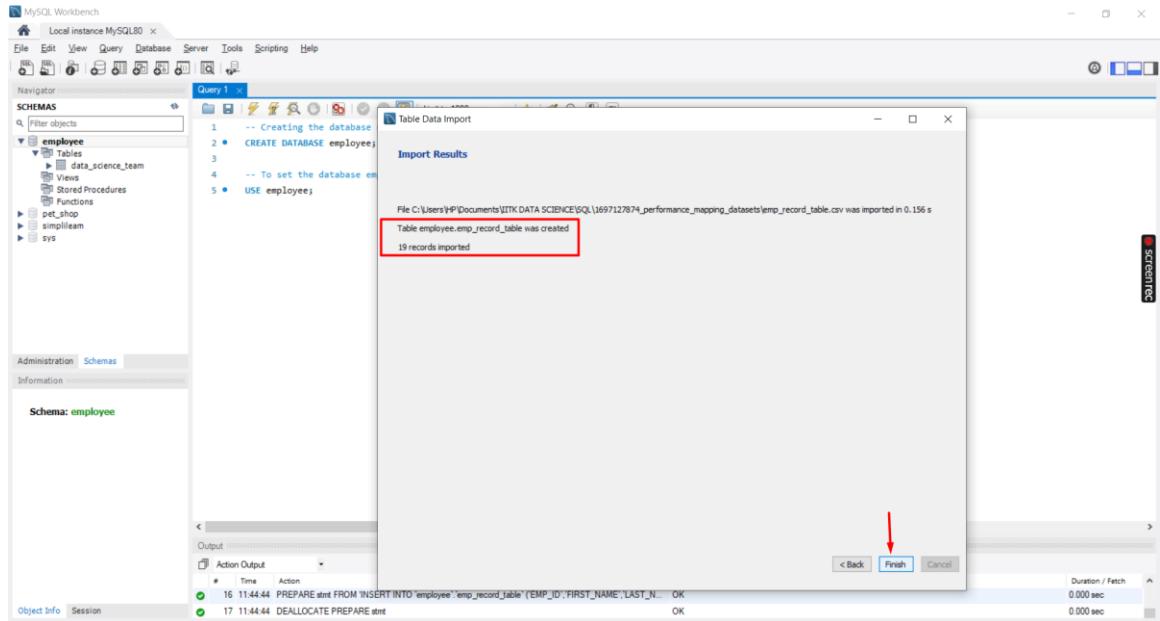
Step 6: In this window, we can see the steps which will be performed while importing the table and click Next which will start importing the table.



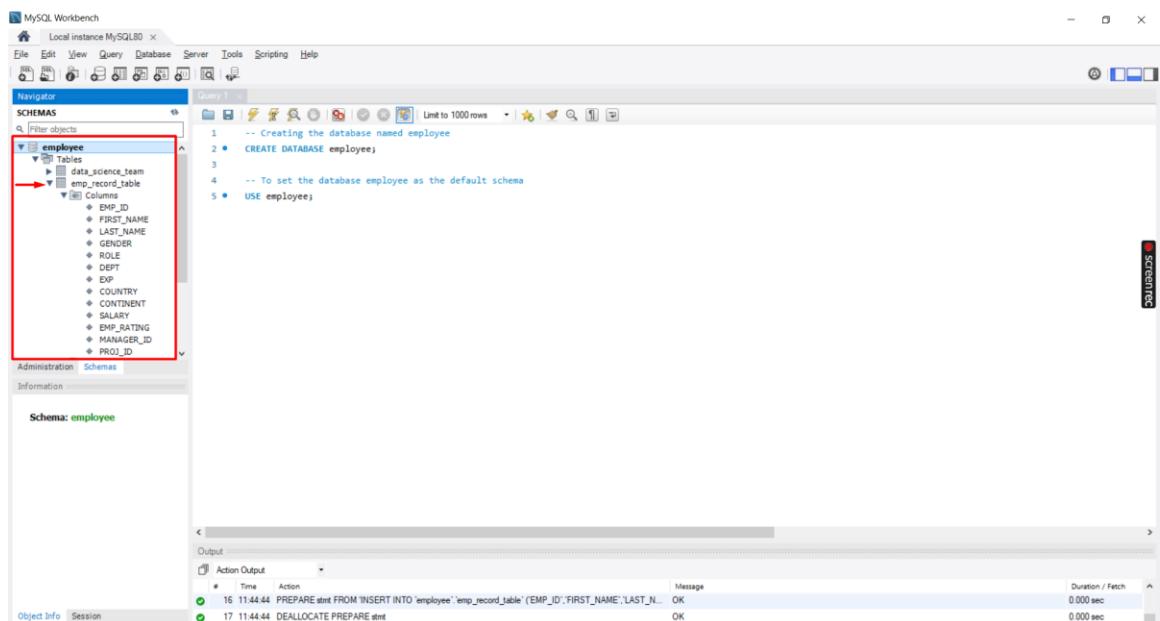
Step 7: Click Next in this window.



Step 8: Here in the below pop up window we can see the table has been created with 19 records. Click Finish to complete the creation of table named emp_record_table.

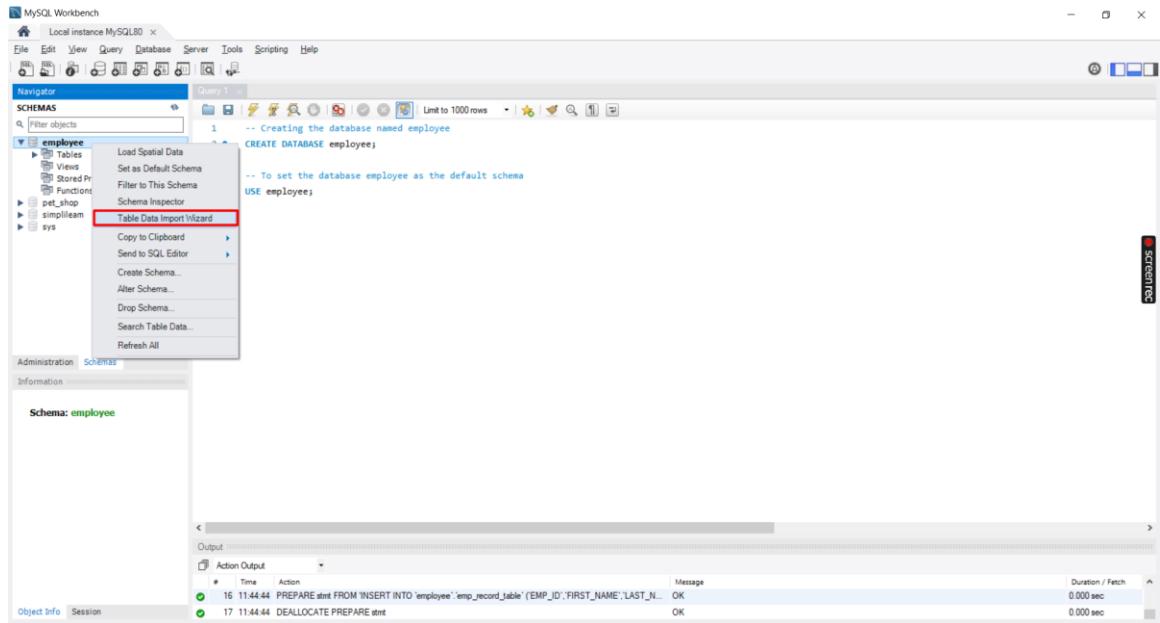


After finishing all the steps we can see the table ‘emp_record_table’ in the database schema ‘employee’.

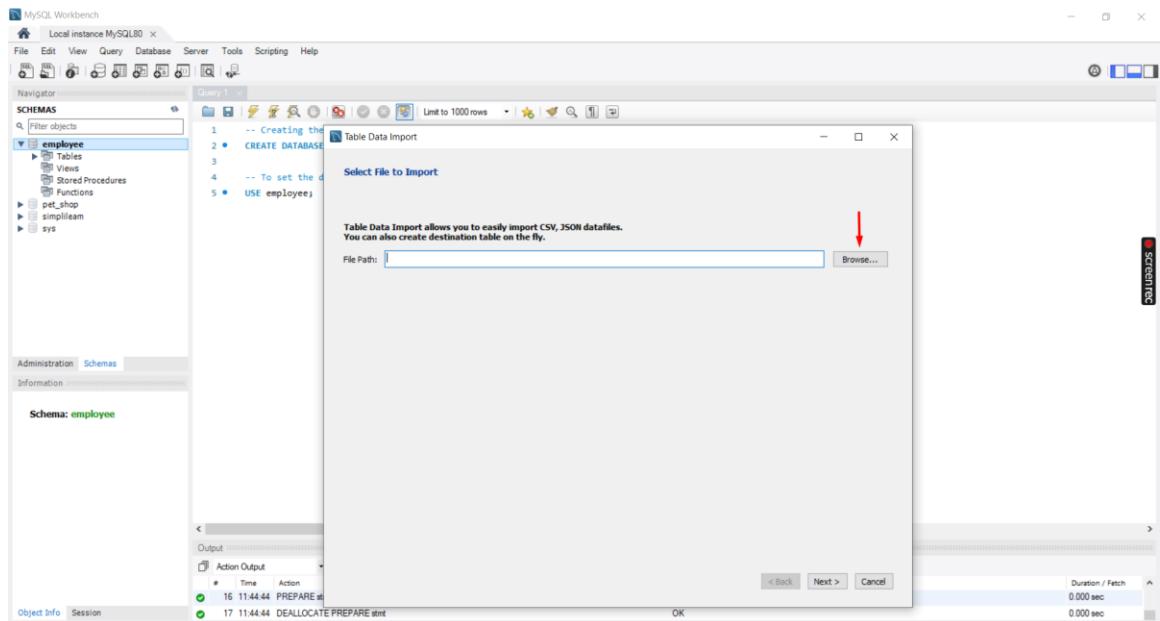


Following are the steps to import the table ‘proj_table’ to the database ‘employee’.

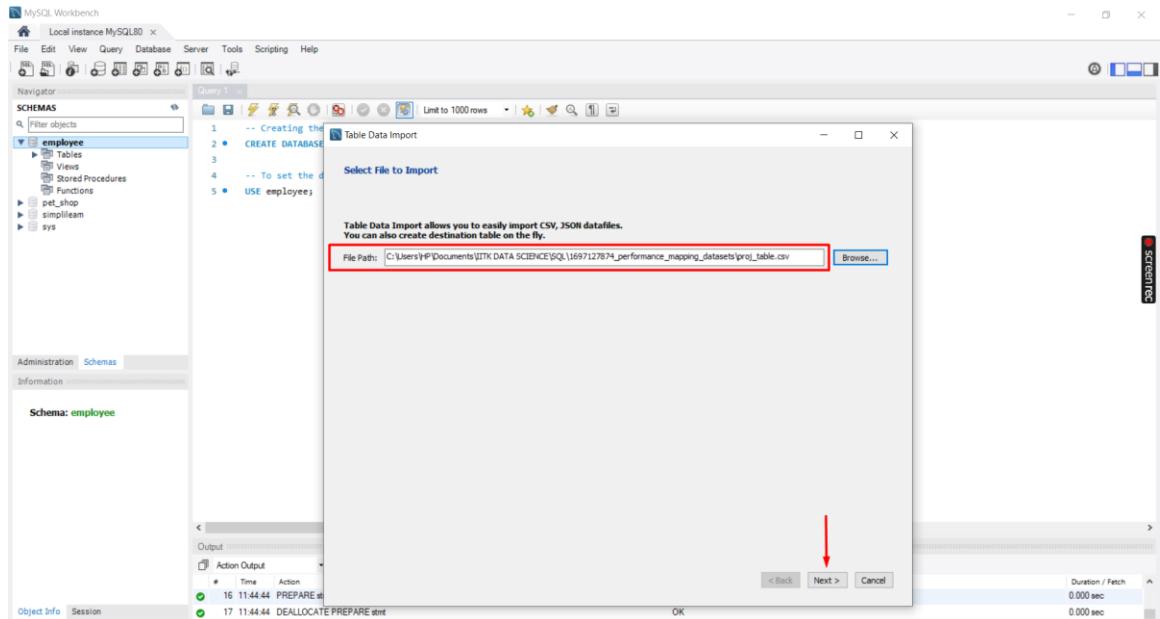
Step 1 : Right click on the employee database and choose ‘Table Data Import Wizard’ as shown below



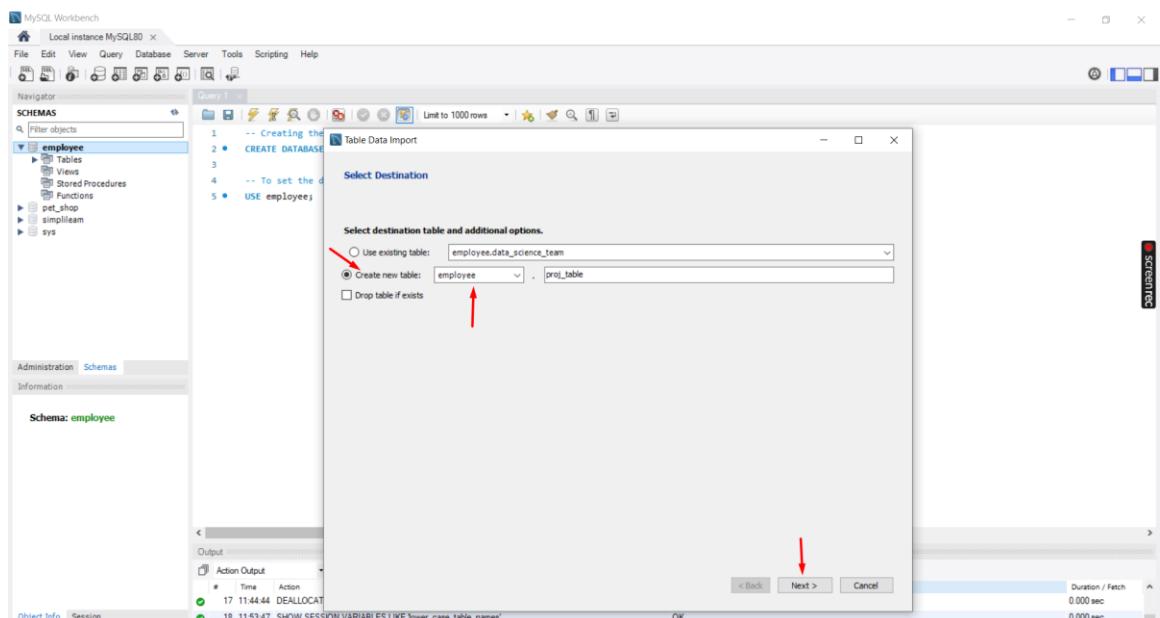
Step 2: Click on ‘Browse’ to select the file proj_table.csv from the right location where the .csv file is saved.



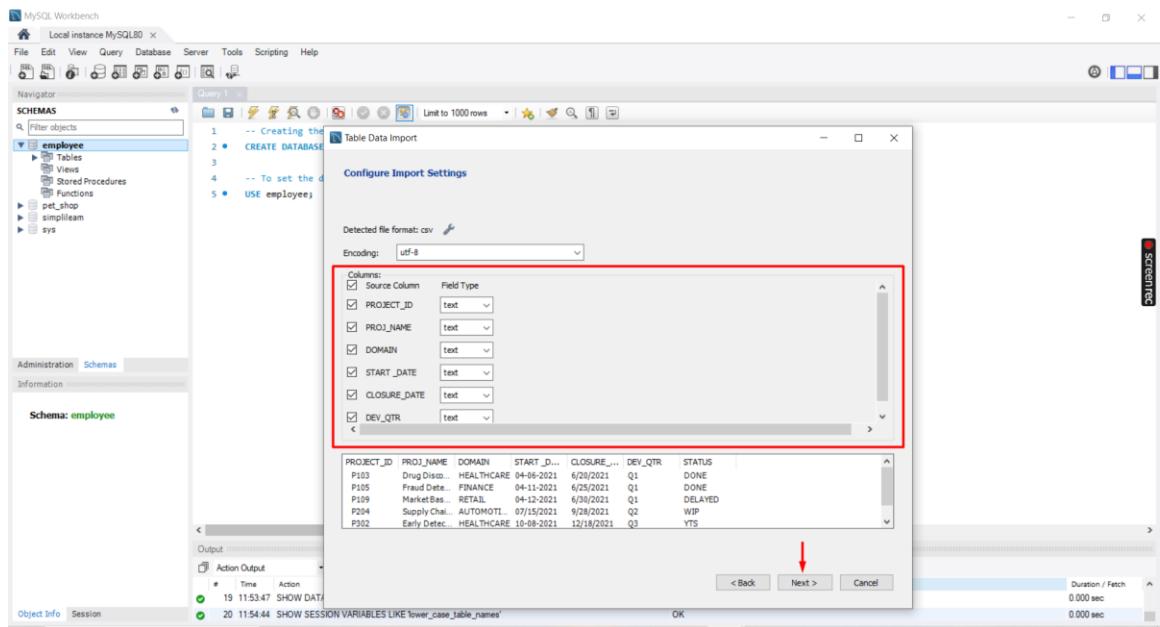
Step 3: Select the file proj_table.csv from the saved location and click Next.



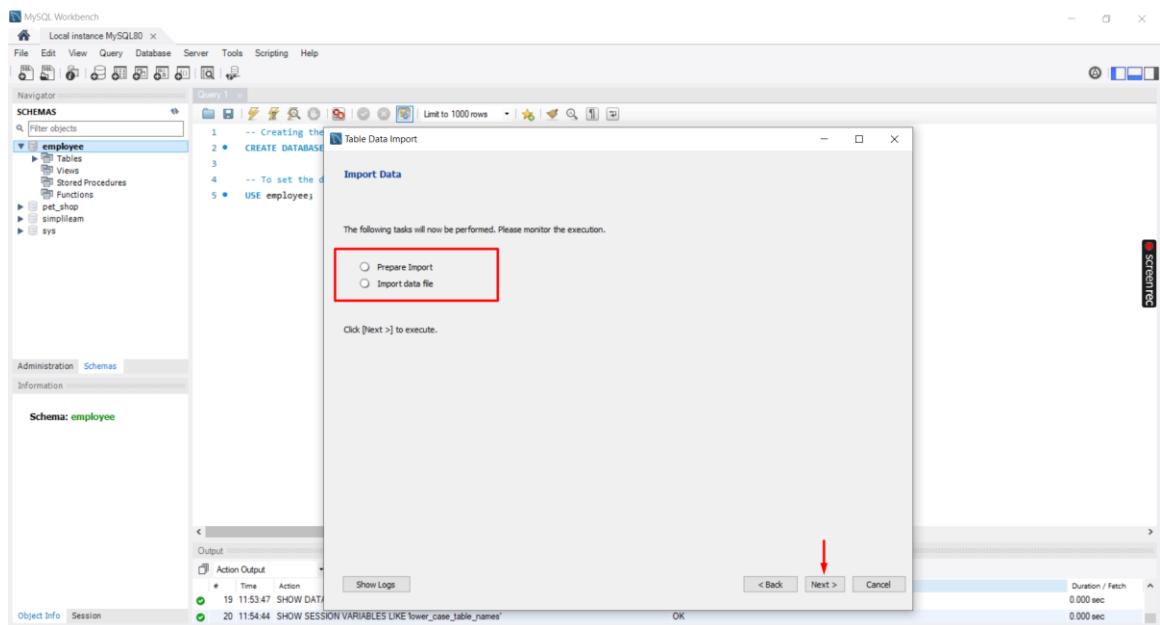
Step 4: Select ‘Create new table’ and make sure that the database is ‘employee’ then click Next.



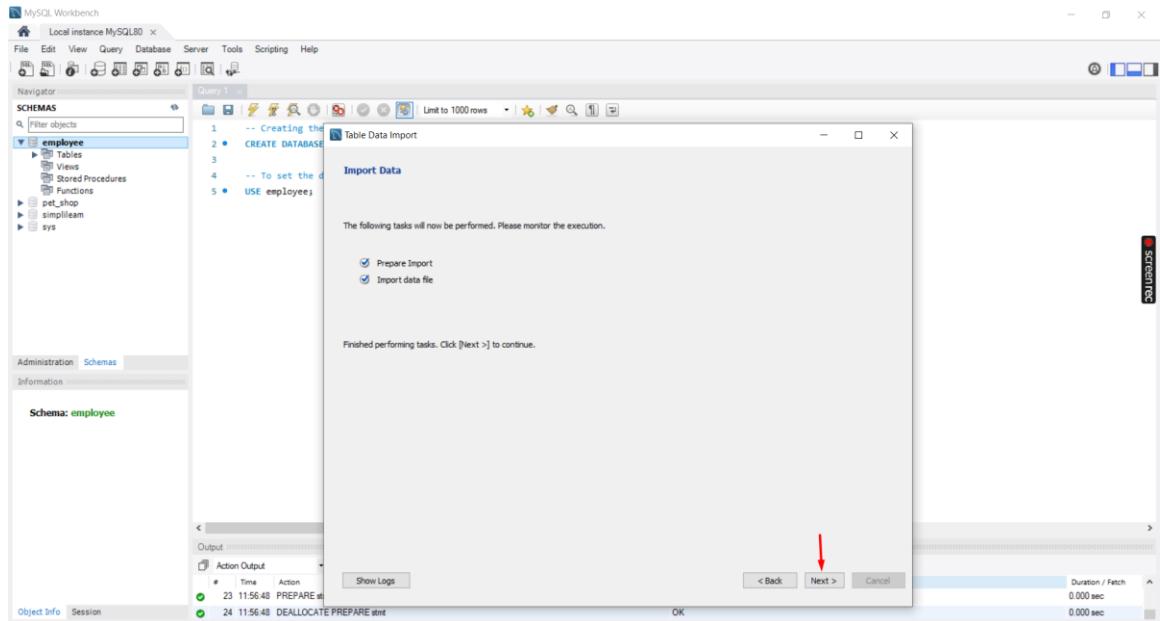
Step 5: Verify all the datatypes from the table which will be imported and click on Next.



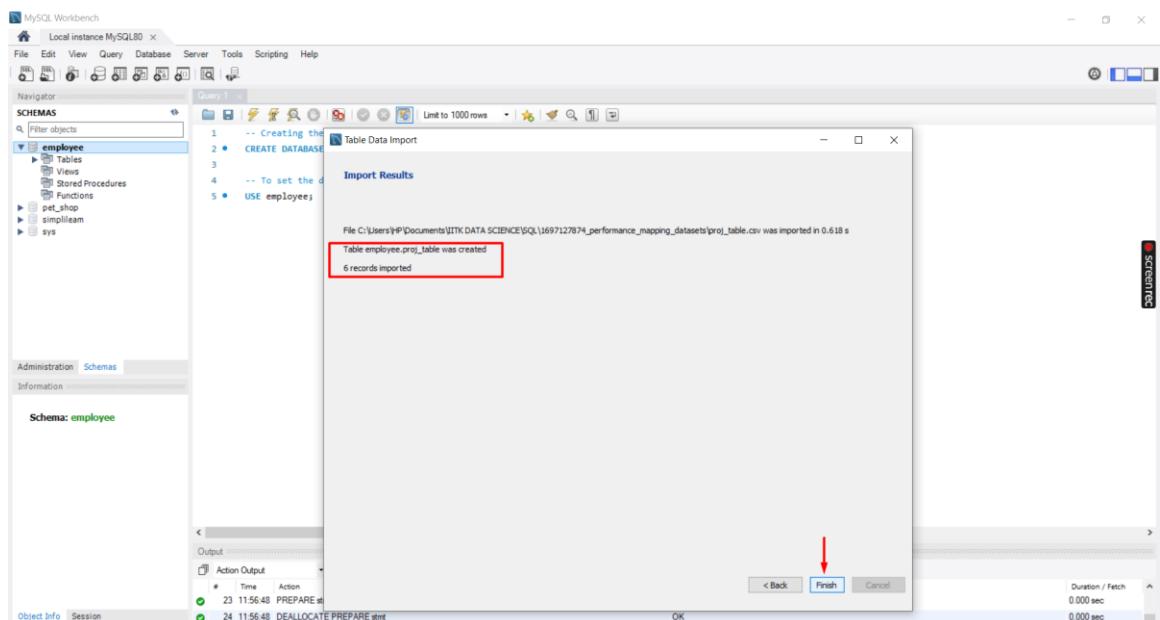
Step 6: In this window, we can see the steps which will be performed while importing the table and click Next which will start importing the table.



Step 7: Click Next in this window.



Step 8: Here in the below pop up window we can see the table has been created with 6 records. Click Finish to complete the creation of table named proj_table.



After finishing all the steps we can see the table ‘proj_table’ in the database schema ‘employee’.

The screenshot shows the MySQL Workbench interface. In the Navigator pane, the 'employee' schema is selected, and the 'Tables' section is expanded, showing 'proj_table'. A red arrow points to this table. The Query Editor window contains the following SQL code:

```

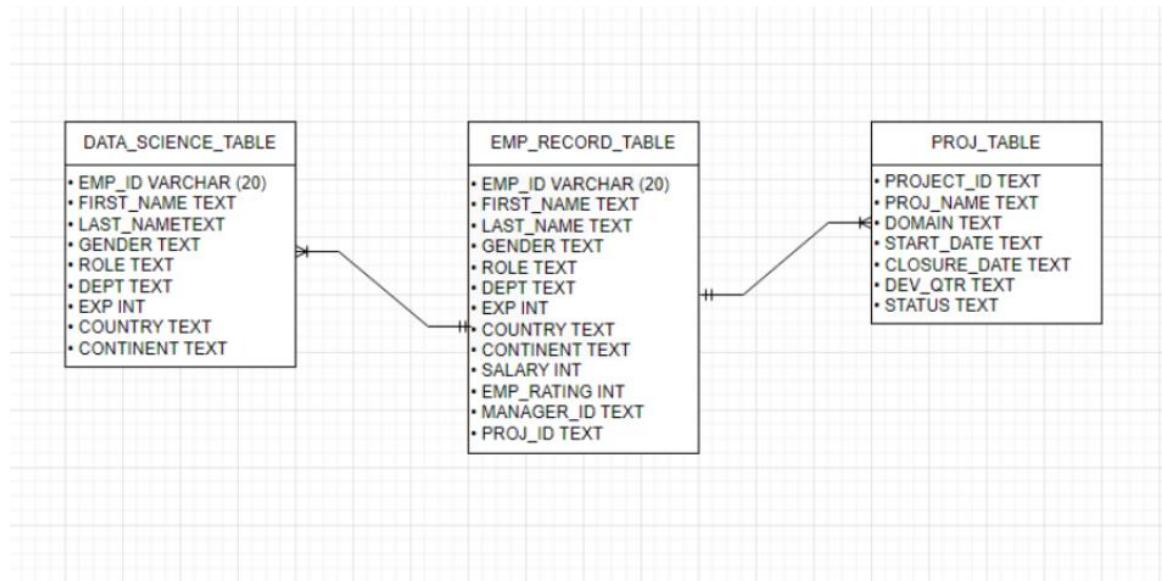
1 -- Creating the database named employee
2 • CREATE DATABASE employee;
3
4 -- To set the database employee as the default schema
5 • USE employee;

```

The Output pane shows the results of the last two statements:

Action	Time	Message	Duration / Fetch
PREPARE stmt FROM INSERT INTO employee`.`proj_table` ('PROJECT_ID','PROJ_NAME','DOMAIN','S...	23 11:56:48	OK	0.000 sec
DEALLOCATE PREPARE stmt	24 11:56:48	OK	0.000 sec

2. Create an ER diagram for the given employee database

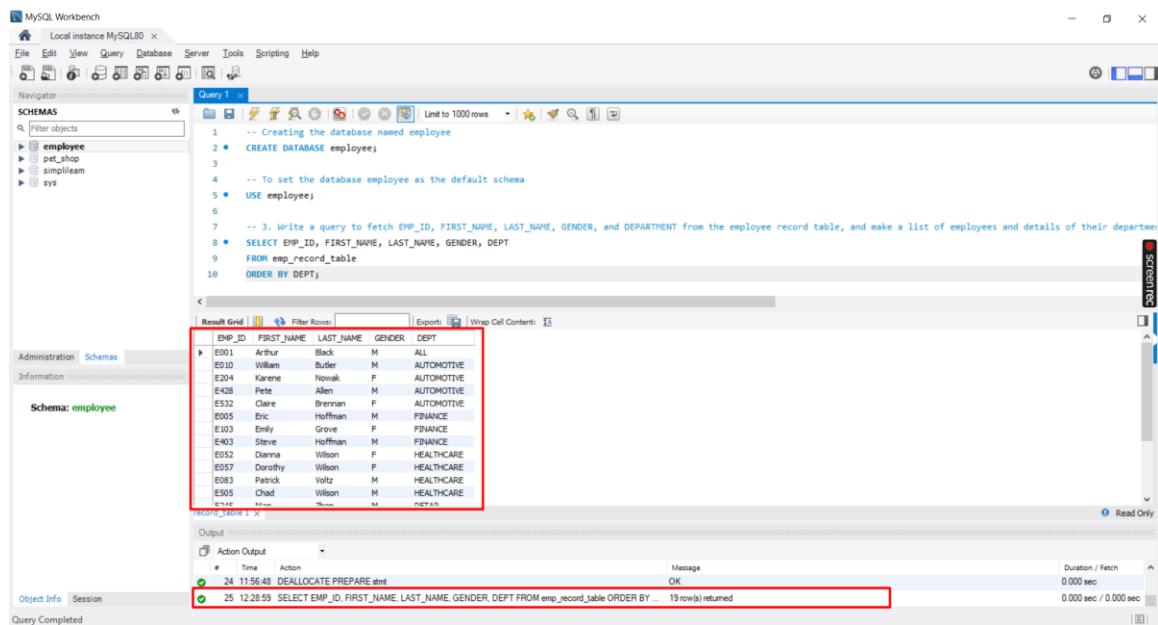


3. Write a query to fetch EMP_ID, FIRST_NAME, LAST_NAME, GENDER, and DEPARTMENT from the employee record table, and make a list of employees and details of their department.

Here we have to retrieve the EMP_ID, FIRST_NAME, LAST_NAME, GENDER, and DEPT from the emp_record_table and group by the DEPT. For that execute the below query:

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

Result:



The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** Contains the SQL query:

```
-- Creating the database named employee
CREATE DATABASE employee;
-- To set the database employee as the default schema
USE employee;
-- 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
SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT
FROM emp_record_table
ORDER BY DEPT;
```
- Result Grid:** Displays the query results in a table format. The columns are EMP_ID, FIRST_NAME, LAST_NAME, GENDER, and DEPT. The data is as follows:

EMP_ID	FIRST_NAME	LAST_NAME	GENDER	DEPT
E001	Arthur	Black	M	ALL
E010	Willam	Butler	M	AUTOMOTIVE
E204	Karenne	Nowak	F	AUTOMOTIVE
E428	Pete	Aller	M	AUTOMOTIVE
E532	Claire	Brennan	F	AUTOMOTIVE
E005	Eric	Hoffman	M	FINANCE
E103	Emily	Grafen	F	FINANCE
E403	Steve	Hoffman	M	FINANCE
E052	Dionna	Wilson	F	HEALTHCARE
E057	Dorothy	Wilson	F	HEALTHCARE
E083	Patrick	Voltz	M	HEALTHCARE
E505	Chad	Wilson	M	HEALTHCARE
E514	John	Wilson	M	HEALTHCARE
- Action Output:** Shows the execution log with two entries:

#	Time	Action	Message	Duration / Fetch
24	11:56:48	DEALLOCATE PREPARE stmt	OK	0.000 sec
25	12:28:59	SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT FROM emp_record_table ORDER BY ...	19 row(s) returned	0.000 sec / 0.000 sec

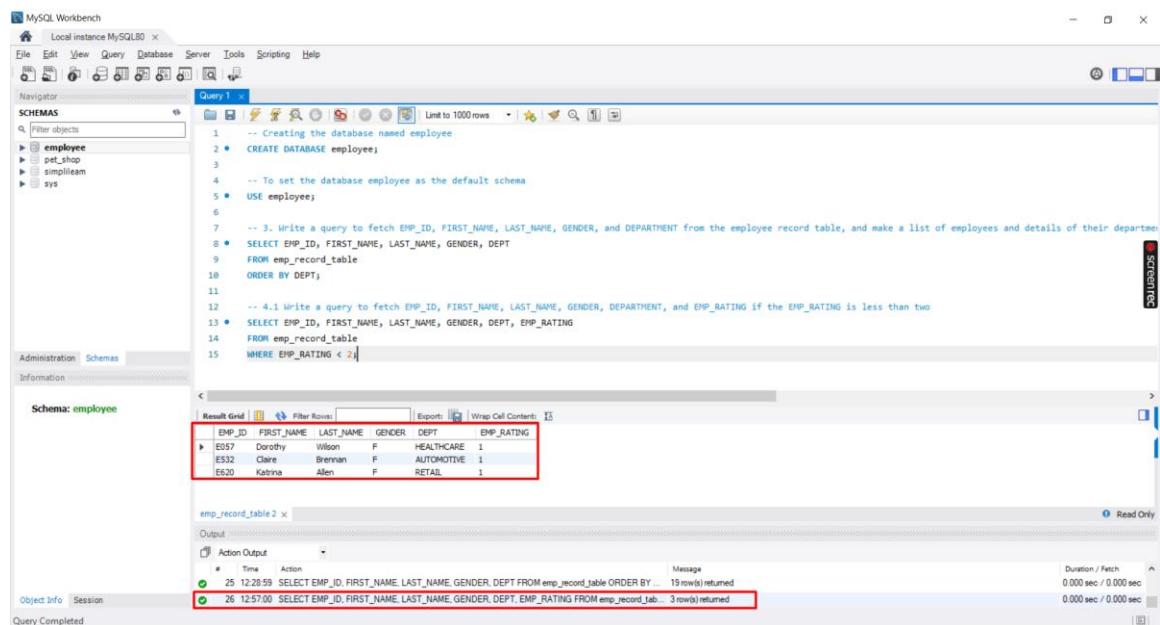
4. Write a query to fetch EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPARTMENT, and EMP_RATING if the EMP_RATING is:
- less than two
 - greater than four
 - between two and four

4.1: If the EMP_RATING is less than two

We can retrieve EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT and EMP_RATING whose EMP_RATING is less than 2 from the emp_record_table using the below query:

```
SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT, EMP_RATING
FROM emp_record_table
WHERE EMP_RATING < 2;
```

Result:



The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** Contains the SQL query:

```
-- Creating the database named employee
CREATE DATABASE employee;
-- To set the database employee as the default schema
USE employee;
-- 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
SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT
FROM emp_record_table
ORDER BY DEPT;
-- 4.1 Write a query to fetch EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPARTMENT, and EMP_RATING if the EMP_RATING is less than two
SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT, EMP_RATING
FROM emp_record_table
WHERE EMP_RATING < 2;
```
- Results Grid:** Shows the output of the last query, displaying three rows of data:

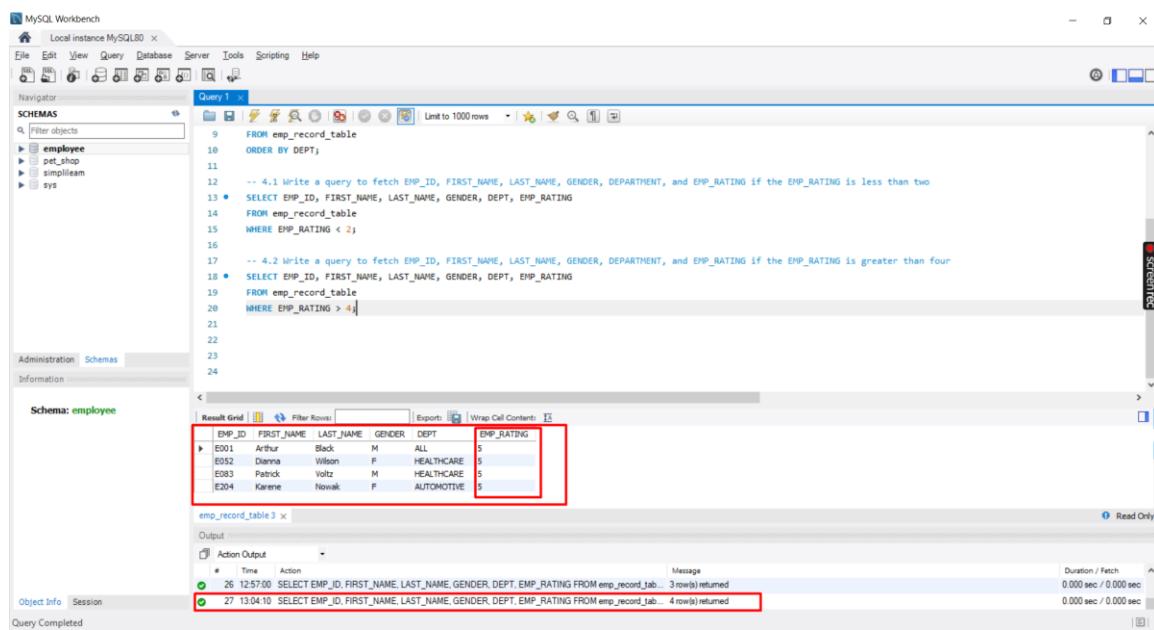
EMP_ID	FIRST_NAME	LAST_NAME	GENDER	DEPT	EMP_RATING
E057	Dorothy	Wilson	F	HEALTHCARE	1
E532	Clare	Brennan	F	AUTOMOTIVE	1
E620	Katrina	Allen	F	RETAIL	1
- Output Panel:** Displays the execution log with two entries:
 - 25 12:28:59 SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT FROM emp_record_table ORDER BY ... 19 row(s) returned
 - 26 12:27:00 SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT, EMP_RATING FROM emp_record_table WHERE EMP_RATING < 2 3 row(s) returned

4.2: If the EMP_RATING is greater than four

We can retrieve EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT and EMP_RATING whose EMP_RATING is greater than four from the emp_record_table using the below query:

```
SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT, EMP_RATING  
FROM emp_record_table  
WHERE EMP_RATING > 4;
```

Result:



The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** employee
- Query Editor:** Contains the SQL query for selecting employees with EMP_RATING > 4.
- Results Grid:** Displays the retrieved data from the emp_record_table. The columns are EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT, and EMP_RATING. The rows are:

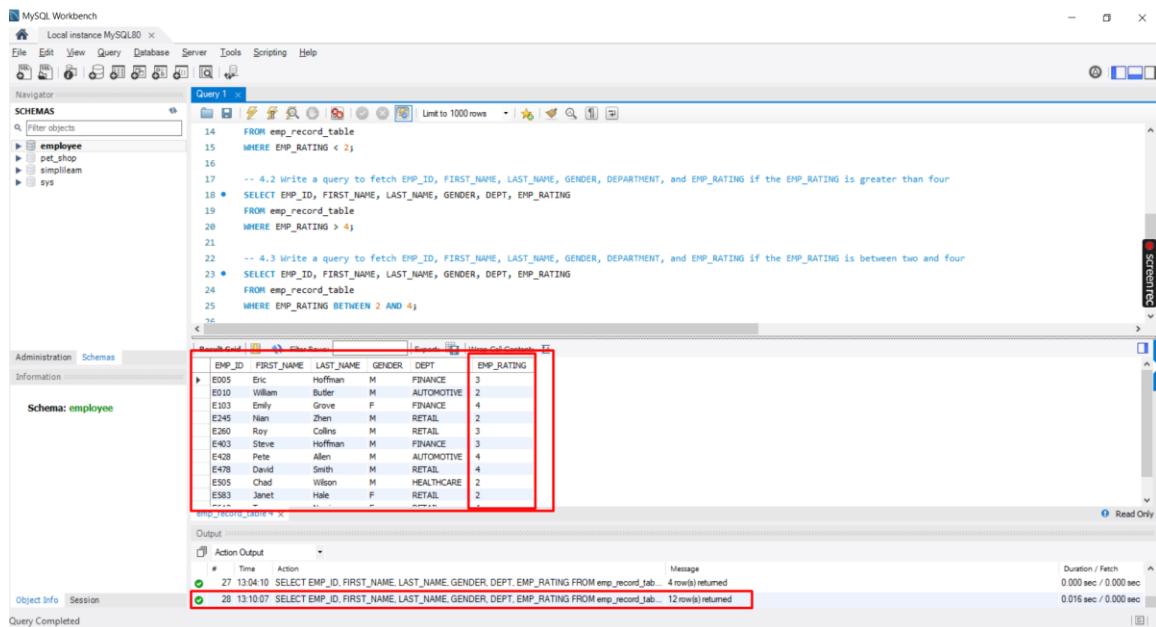
EMP_ID	FIRST_NAME	LAST_NAME	GENDER	DEPT	EMP_RATING
E001	Arthur	Black	M	ALL	5
E052	Dianna	Wilson	F	HEALTHCARE	5
E083	Patrick	Volz	M	HEALTHCARE	5
E204	Karen	Nowak	F	AUTOMOTIVE	5
- Output:** Shows the execution log with two entries, both of which completed successfully with 0.000 sec / 0.000 sec duration.

4.3: If the EMP_RATING is between two and four

We can retrieve EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT and EMP_RATING whose EMP_RATING is between two and four from the emp_record_table using the below query:

```
SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT, EMP_RATING  
FROM emp_record_table  
WHERE EMP_RATING BETWEEN 2 AND 4;
```

Result:



The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the schema **employee** selected.
- Query Editor:** Displays the SQL query:

```
14  FROM emp_record_table  
15  WHERE EMP_RATING < 2;  
16  
17  -- 4.2 Write a query to fetch EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPARTMENT, and EMP_RATING if the EMP_RATING is greater than four  
18 •  SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT, EMP_RATING  
19  FROM emp_record_table  
20  WHERE EMP_RATING > 4;  
21  
22  -- 4.3 Write a query to fetch EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPARTMENT, and EMP_RATING if the EMP_RATING is between two and four  
23 •  SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT, EMP_RATING  
24  FROM emp_record_table  
25  WHERE EMP_RATING BETWEEN 2 AND 4;
```
- Results Grid:** Shows the results of the query with EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT, and EMP_RATING columns. The results are:

EMP_ID	FIRST_NAME	LAST_NAME	GENDER	DEPT	EMP_RATING
E005	Eric	Hoffman	M	FINANCE	3
E010	William	Butler	M	AUTOMOTIVE	2
E103	Emily	Grove	F	FINANCE	4
E245	Niall	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
E583	Janet	Hale	F	RETAIL	2
- Output:** Shows the execution log with two entries:

#	Time	Action	Message	Duration / Fetch
27	13:04:10	SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT, EMP_RATING FROM emp_record_table...	4 row(s) returned	0.000 sec / 0.000 sec
28	13:10:07	SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT, EMP_RATING FROM emp_record_table...	12 row(s) returned	0.016 sec / 0.000 sec

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.

Here we have to concatenate the FIRST_NAME and LAST_NAME from the emp_record_table whose department is Finance and give the name of the table as NAME. We have to execute the below query:

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

Result:

The screenshot shows the MySQL Workbench interface. In the top-left pane, the Navigator displays the database schema with the 'employee' schema selected. The main pane contains a 'Query 1' window with the following SQL code:

```
23 • SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT, EMP_RATING  
24   FROM emp_record_table  
25   WHERE EMP_RATING BETWEEN 2 AND 4;  
26  
27 -- 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  
28 • SELECT CONCAT(FIRST_NAME, ' ', LAST_NAME) AS NAME  
29   FROM emp_record_table  
30   WHERE DEPT = 'FINANCE';  
31  
32  
33  
34
```

The 'Result Grid' tab shows the output of the concatenated names:

NAME
Eric Hoffman
Emily Grove
Steve Hoffman

The 'Result 5' tab shows the execution log:

Action Output	Time	Action	Message	Duration / Fetch
28	13:10:07	SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT, EMP_RATING FROM emp_record_table...	12 row(s) returned	0.015 sec / 0.000 sec
29	13:51:40	SELECT CONCAT(FIRST_NAME, ' ', LAST_NAME) AS NAME FROM emp_record_table WHERE DEPT = 'FINANCE'	3 row(s) returned	0.015 sec / 0.000 sec

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

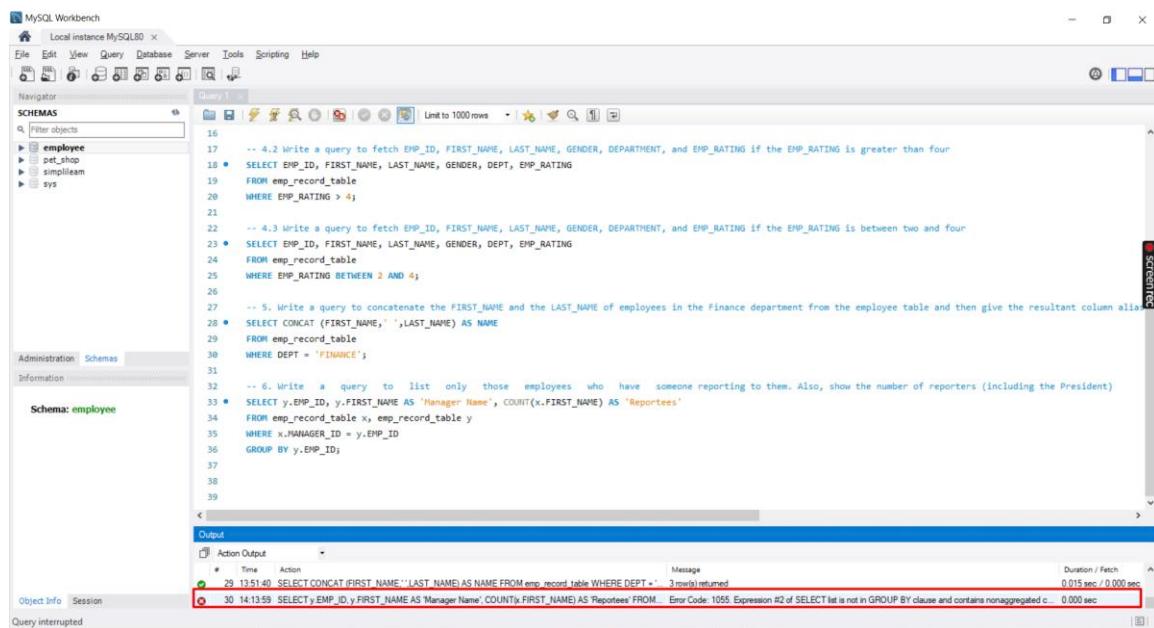
Here we have to retrieve the employees who have someone reporting to them and also showing the number of reportees (including the President). So the query is:

```
SELECT y.EMP_ID,
       y.FIRST_NAME AS 'Manager Name',
       COUNT(x.FIRST_NAME) AS 'Reportees'
  FROM emp_record_table x, emp_record_table y
 WHERE x.MANAGER_ID = y.EMP_ID
 GROUP BY y.EMP_ID;
```

Error Handling:

When the above query is executed we get the error as below:

“Error Code: 1055. Expression #2 of SELECT list is not in GROUP BY clause and contains nonaggregated column ‘employee.y.FIRST_NAME’ which is not functionally dependent on columns in GROUP BY clause; this is incompatible with sql_mode=only_full_group_by 0.000 sec”



The screenshot shows the MySQL Workbench interface with a query editor and output window. The query editor contains the SQL code for question 6. The output window shows the execution of the query. The last two rows of the output are highlighted with red boxes, indicating the error. The error message is: "30 14:13:59 SELECT y.EMP_ID, y.FIRST_NAME AS 'Manager Name', COUNT(x.FIRST_NAME) AS 'Reportees' FROM emp_record_table x, emp_record_table y WHERE x.MANAGER_ID = y.EMP_ID GROUP BY y.EMP_ID; Error Code: 1055. Expression #2 of SELECT list is not in GROUP BY clause and contains nonaggregated column 'employee.y.FIRST_NAME' which is not functionally dependent on columns in GROUP BY clause; this is incompatible with sql_mode=only_full_group_by 0.000 sec".

Error code 1055 occurs when we include a column in the SELECT list and we omit it from the GROUP BY clause. This is because our sql_mode contains ONLY_FULL_GROUP_BY enabled.

To check the sql_mode, execute the below query:

```
SELECT @@sql_mode;
```

Output of the above query:

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (employee, pet_shop, simplilearn, sys) and Administration (Schemas, Information).
- Query Editor:** Title: Query 1. Text area contains a multi-line query. Line 38 is highlighted with a red box and contains the statement `SELECT @@sql_mode`. The output pane below shows the result of this query as `@@sql_mode`.
- Output Pane:** Result Grid, Filter Rows, Export, Wrap Cell Content. It displays the value of `@@sql_mode` which is `ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION`.
- Status Bar:** Result 6 x, Read On Click.
- Log:** Shows two entries:
 - Line 30: Error Code: 1055. Expression #2 of SELECT list is not in GROUP BY clause and contains nonaggregated column 'emp_id' specified in select list. The error occurred at line 38.
 - Line 31: Success: SELECT @sql_mode LIMIT 0, 1000. 1 row(s) returned.
- Object Info:** Session.

In order to disable the ONLY_FULL_GROUP_BY mode execute the below query:

MySQL Workbench - Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

- employee
- pet_shop
- simplilearn
- sys

Query 1

```
21 -- 4.3 Write a query to fetch EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPARTMENT, and EMP_RATING if the EMP_RATING is between two and four
22 • SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT, EMP_RATING
23   FROM emp_record_table
24   WHERE EMP_RATING BETWEEN 2 AND 4;
25
26
27 -- 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
28 • SELECT CONCAT(FIRST_NAME, ' ', LAST_NAME) AS NAME
29   FROM emp_record_table
30   WHERE DEPT = 'FINANCE';
31
32 -- 6. Write a query to list only those employees who have someone reporting to them. Also, show the number of reporters (including the President)
33 • SELECT y.EMP_ID, y.FIRST_NAME AS 'Manager Name', COUNT(x.FIRST_NAME) AS 'Reportees'
34   FROM emp_record_table x, emp_record_table y
35   WHERE x.MANAGER_ID = y.EMP_ID
36   GROUP BY y.EMP_ID;
37
38 • SELECT @@sql_mode;
39
40 • SET @@sql_mode = SYS_LIST_DROP(@@sql_mode, 'ONLY_FULL_GROUP_BY');
41
42
43
44
```

Action Output

#	Time	Action	Message	Duration / Fetch
31	14:22:35	SELECT @@sql_mode LIMIT 0,1000	1 row(s) returned	0.000 sec / 0.000 sec
32	14:26:09	SET @@sql_mode = SYS_LIST_DROP(@@sql_mode, 'ONLY_FULL_GROUP_BY')	0 row(s) affected	0.000 sec

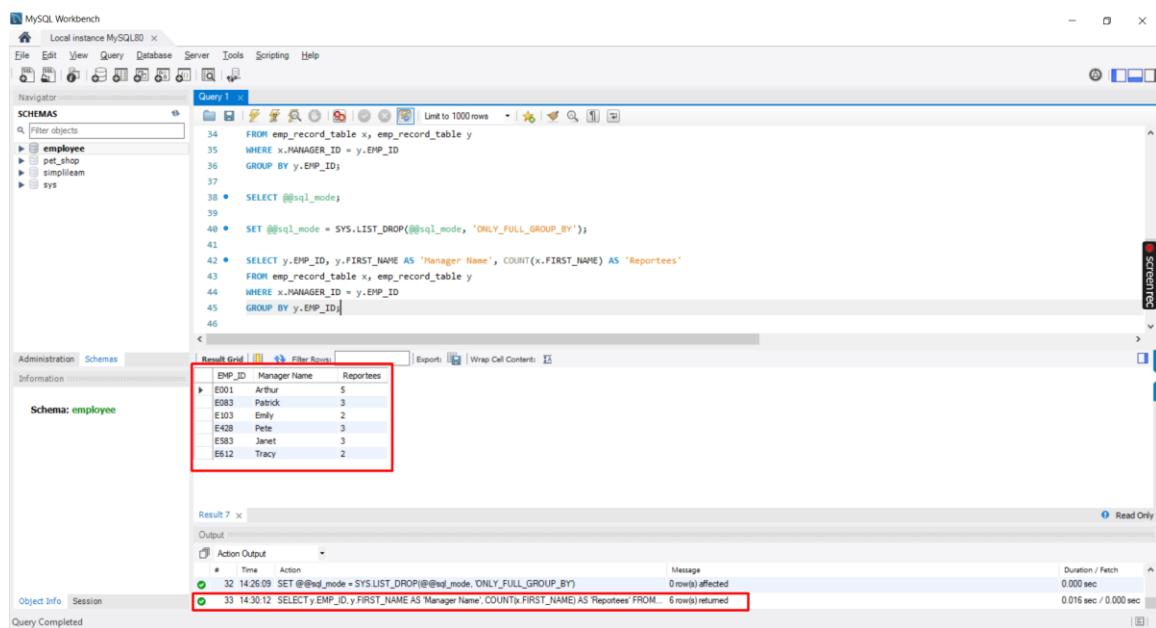
Object Info Session

Query Completed

Now we can execute the below query to see the result of the employees who have someone reporting to them and also showing the number of reportees (including the President).

```
SELECT y.EMP_ID,
       y.FIRST_NAME AS 'Manager Name',
       COUNT(x.FIRST_NAME) AS 'Reportees'
  FROM emp_record_table x, emp_record_table y
 WHERE x.MANAGER_ID = y.EMP_ID
 GROUP BY y.EMP_ID;
```

Result:



The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** Contains the SQL query provided above.
- Results Grid:** Displays the output of the query, showing the Manager ID, Manager Name, and Reportees count for each employee. The results are:

EMP_ID	Manager Name	Reportees
E001	Arthur	5
E093	Patrick	3
E103	Billy	3
E428	Pete	3
E583	Janet	3
E512	Tracy	2

- Output Tab:** Shows the execution log with two entries. The second entry is highlighted with a red box:

 - Time: 32 14:26:09 Action: SET @sql_mode = SYS_LIST_DROP(@@sql_mode, 'ONLY_FULL_GROUP_BY')
 - Time: 33 14:30:12 Action: SELECT y.EMP_ID, y.FIRST_NAME AS 'Manager Name', COUNT(x.FIRST_NAME) AS 'Reportees' FROM emp_record_table x, emp_record_table y WHERE x.MANAGER_ID = y.EMP_ID GROUP BY y.EMP_ID

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.

We have to list all the employees from both healthcare and finance departments using UNION clause. So the query is:

```
SELECT
    EMP_ID,FIRST_NAME,DEPT
FROM emp_record_table
WHERE DEPT = 'HEALTHCARE'
UNION
SELECT
    EMP_ID,FIRST_NAME,DEPT
FROM emp_record_table
WHERE DEPT = 'FINANCE';
```

Result:

The screenshot shows the MySQL Workbench interface with a query editor and a results grid. The query editor contains the SQL code provided above. The results grid displays the output of the query, which lists employees from the 'employee' table categorized by department ('HEALTHCARE' and 'FINANCE'). The results are as follows:

EMP_ID	FIRST_NAME	DEPT
E052	Diana	HEALTHCARE
E057	Dorothy	HEALTHCARE
E083	Patrick	HEALTHCARE
E050	Chad	HEALTHCARE
E005	Erik	FINANCE
E103	Emily	FINANCE
E403	Steve	FINANCE

The status bar at the bottom indicates the query was completed successfully.

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.

We have to retrieve the employee details such as EMP_ID, FIRST_NAME, LAST_NAME, ROLE, DEPARTMENT, and EMP_RATING from the emp_record_table with the conditions such as

1. Group by department
2. Including the respective employee rating
3. The maximum emp_rating for the department

So the query that should be executed will be:

```
SELECT
    EMP_ID,
    FIRST_NAME,
    LAST_NAME,
    ROLE,
    DEPT,
    EMP_RATING,
    MAX(EMP_RATING) OVER (PARTITION BY DEPT)
        AS 'Max Employee Rating/Department'
FROM emp_record_table;
```

Result:

The screenshot shows the MySQL Workbench interface with the following details:

- File Bar:** Local instance MySQL80, File, Edit, View, Query, Database, Server, Tools, Scripting, Help.
- Navigator:** Schemas (employee, pet_shop, simpleam, sys).
- Query Editor:** Query 1, containing the SQL query provided above.
- Results Grid:** A table showing employee data with an additional column for 'Max Employee Rating/Department'. The table has columns: EMP_ID, FIRST_NAME, LAST_NAME, ROLE, DEPT, EMP_RATING, and Max Employee Rating/Department. The data is as follows:

EMP_ID	FIRST_NAME	LAST_NAME	ROLE	DEPT	EMP_RATING	Max Employee Rating/Department
E001	Arthur	Black	PRESIDENT	ALL	5	5
E010	William	Butler	LEAD DATA SCIENTIST	AUTOMOTIVE	2	5
E024	Karen	Nowak	SENIOR DATA SCIENTIST	AUTOMOTIVE	5	5
E428	Pete	Allen	MANAGER	AUTOMOTIVE	4	5
E532	Clare	Brennan	ASSOCIATE DATA SCIENTIST	AUTOMOTIVE	1	5
E545	Eric	Feuerstein	DATA SCIENTIST	FINANCE	5	4
E503	Emily	Grove	MANAGER	FINANCE	4	4
E403	Steve	Hoffman	ASSOCIATE DATA SCIENTIST	FINANCE	3	4
E552	Diana	Wilson	SENIOR DATA SCIENTIST	HEALTHCARE	5	5
E557	Dorothy	Wilson	SENIOR DATA SCIENTIST	HEALTHCARE	1	5

- Output Window:** Shows the execution log with three rows of output.

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.

We have to retrieve the EMP_ID, FIRST_NAME, LAST_NAME, ROLE, DEPT, and SALARY of each employee, the minimum and maximum salary for that particular role in the company. So the query to be executed will be:

```

SELECT
    EMP_ID,FIRST_NAME, LAST_NAME,
    DEPT,ROLE,SALARY,
    MIN(SALARY) OVER(PARTITION BY ROLE)
        AS 'Minimum Salary',
    MAX(SALARY) OVER(PARTITION BY ROLE)
        AS 'Maximum Salary'
FROM emp_record_table;

```

Result:

The screenshot shows the MySQL Workbench interface with the following details:

- Navigator:** Shows the Schemas: employee, pet_shop, simpleteam, sys.
- Query Editor:** Contains the SQL query:

```

69 -- 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.
70 • SELECT
71     EMP_ID,
72     FIRST_NAME,
73     LAST_NAME,
74     DEPT,
75     ROLE,
76     SALARY,
77     MIN(SALARY) OVER(PARTITION BY ROLE) AS 'Minimum Salary',
78     MAX(SALARY) OVER(PARTITION BY ROLE) AS 'Maximum Salary'
79     FROM emp_record_table;
80

```
- Results Grid:** Displays the query results in a table format:

EMP_ID	FIRST_NAME	LAST_NAME	DEPT	ROLE	SALARY	Minimum Salary	Maximum Salary
E003	Patrick	Volz	HEALTHCARE	MANAGER	9500	8500	11000
E004	Elroy	Gyre	PRIVACY	PRESIDENT	16500	16500	16500
E429	Pete	Allen	AUTOMOTIVE	MANAGER	11000	8500	11000
E583	Janet	Hale	RETAIL	MANAGER	10000	8500	11000
E512	Tracy	Norris	RETAIL	MANAGER	8500	8500	11000
E001	Arthur	Black	ALL	PRESIDENT	16500	16500	16500
E052	Dianna	Wilson	HEALTHCARE	SENIOR DATA SCIENTIST	5500	5500	7700
E057	Dorothy	Wilson	HEALTHCARE	SENIOR DATA SCIENTIST	7700	5500	7700
E204	Karen	Nowak	AUTOMOTIVE	SENIOR DATA SCIENTIST	7500	5500	7700
E245	Nan	Zhen	RETAIL	SENIOR DATA SCIENTIST	6500	5500	7700
E260	Roy	Collins	RETAIL	SENIOR DATA SCIENTIST	7000	5500	7700
- Output:** Shows the execution log with two entries:

```

35 14:56:06 SELECT EMP_ID, FIRST_NAME, LAST_NAME, ROLE, DEPT, EMP_RATING, - Employee ... 19 row(s) returned
36 15:14:53 SELECT EMP_ID, FIRST_NAME, LAST_NAME, DEPT, ROLE, SALARY, MIN(SALARY)... 19 row(s) returned

```
- Object Info:** Shows the session information.

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

Here we have to assign ranks to each employee based on their experience and dense rank based on experience. The query to be executed will be:

```
SELECT  
    EMP_ID, FIRST_NAME, LAST_NAME,  
    ROLE, DEPT, EXP,  
    RANK() OVER(ORDER BY EXP DESC)  
        AS 'Employee Experience Rank',  
    DENSE_RANK() OVER(ORDER BY EXP DESC)  
        AS 'Employee Experience Dense Rank'  
FROM emp_record_table;
```

Result:

The screenshot shows the MySQL Workbench interface with a query editor window. The query is:

```
-- 10. Write a query to assign ranks to each employee based on their experience. Take data from the employee record table.  
SELECT  
    EMP_ID, FIRST_NAME, LAST_NAME,  
    ROLE, DEPT, EXP,  
    RANK() OVER(ORDER BY EXP DESC) AS 'Employee Experience Rank',  
    DENSE_RANK() OVER(ORDER BY EXP DESC) AS 'Employee Experience Dense Rank'  
FROM emp_record_table;
```

The results grid displays 19 rows of employee data with calculated ranks. The columns are: EMP_ID, FIRST_NAME, LAST_NAME, ROLE, DEPT, EXP, Employee Experience Rank, and Employee Experience Dense Rank. The data includes various roles like PRESIDENT, MANAGER, and SENIOR DATA SCIENTIST across different departments like HEALTHCARE, FINANCE, AUTOMOTIVE, RETAIL, and LEAD DATA SCIENTIST.

EMP_ID	FIRST_NAME	LAST_NAME	ROLE	DEPT	EXP	Employee Experience Rank	Employee Experience Dense Rank
E001	Arthur	Black	PRESIDENT	ALL	20	1	1
E002	Patrick	Joltz	MANAGER	HEALTHCARE	15	2	2
E003	Eduardo	Guvino	MANAGER	FINANCE	17	3	3
E008	Pete	Afern	MANAGER	AUTOMOTIVE	14	3	3
E009	Janet	Hale	MANAGER	RETAIL	14	3	3
E012	Tracy	Norris	MANAGER	RETAIL	13	6	4
E010	William	Butler	LEAD DATA SCIENTIST	AUTOMOTIVE	12	7	5
E005	Eric	Hoffman	LEAD DATA SCIENTIST	FINANCE	11	8	6
E057	Dorothy	Wilson	SENIOR DATA SCIENTIST	HEALTHCARE	9	9	7
E004	Karen	Nowak	SENIOR DATA SCIENTIST	AUTOMOTIVE	8	10	8
E060	Roy	Collins	SENIOR DATA SCIENTIST	RETAIL	7	11	9
E052	Dianne	Wilson	SENIOR DATA SCIENTIST	HEALTHCARE	6	12	10
E245	Nan	Zhen	SENIOR DATA SCIENTIST	RETAIL	6	12	10
E005	Chad	Wilson	ASSOCIATE DATA SCIENTIST	HEALTHCARE	5	14	11
E003	Steve	Hoffman	ASSOCIATE DATA SCIENTIST	FINANCE	4	15	12

The output pane shows two log entries:

- 36 15:14:53 SELECT EMP_ID, FIRST_NAME, LAST_NAME, DEPT, ROLE, SALARY, MIN(SALARY) 19 row(s) returned
- 37 15:31:45 SELECT EMP_ID, FIRST_NAME, LAST_NAME, ROLE, DEPT, EXP, RANK() OVER(ORDER BY EXP DESC) 19 row(s) returned

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.

We have to create a view that displays employees in various countries whose salary is more than six thousand. The query to be executed will be:

```
CREATE VIEW emp AS  
SELECT * FROM emp_record_table  
WHERE SALARY > 6000;
```

The screenshot shows the MySQL Workbench interface. In the left sidebar, under 'Schemas', the 'employee' schema is selected. The main pane contains the SQL code for creating the view:

```
-- 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.  
CREATE VIEW emp  
AS  
SELECT * FROM emp_record_table  
WHERE SALARY > 6000;
```

The 'Output' pane at the bottom shows the results of the query execution:

Action	Time	Message	Duration / Fetch
37 15:31:45	SELECT EMP_ID, FIRST_NAME, LAST_NAME, ROLE, DEPT, EXP, RANK() OVER(ORDER BY EXP DESC) AS 'Employee Experience Rank', DENSE_RANK() OVER(ORDER BY EXP DESC) AS 'Employee Experience Dense Rank'	19 row(s) returned	0.000 sec / 0.000 sec
38 15:41:14	CREATE VIEW emp AS SELECT * FROM emp_record_table WHERE SALARY > 6000	0 row(s) affected	0.015 sec

Now we have created the view. Now with the below query we have to list of all employees from various countries whose salary is more than six thousand

```
SELECT * FROM emp;
```

Result:

The screenshot shows the MySQL Workbench interface with a query editor and results grid.

Query Editor (Query 1):

```

89 -- 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.
90 • CREATE VIEW emp
91     AS
92         SELECT * FROM emp_record_table
93         WHERE SALARY > 6000;
94
95 • SELECT * FROM emp;
96
97
98

```

Result Grid:

EMP_ID	FIRST_NAME	LAST_NAME	GENDER	ROLE	DEPT	EXP	COUNTRY	CONTINENT	SALARY	EMP_RATING	MANAGER_ID	PROJ_ID
E001	Arthur	Blak	M	PRESIDENT	ALL	20	USA	NORTH AMERICA	16500	5	E103	P105
E002	Eric	Hoffman	M	SENIOR DATA SCIENTIST	FINANCE	11	USA	NORTH AMERICA	8000	3	E428	P104
E010	William	Bulter	M	LEAD DATA SCIENTIST	AUTOMOTIVE	12	FRANCE	EUROPE	7000	2	E583	P202
E037	Dorothy	Wifler	F	SENIOR DATA SCIENTIST	HEALTHCARE	9	USA	NORTH AMERICA	7700	3	E001	P103
E083	Patrick	Volz	M	MANAGER	HEALTHCARE	15	USA	NORTH AMERICA	9500	5	E001	P103
E103	Emily	Grove	F	MANAGER	FINANCE	14	CANADA	NORTH AMERICA	10500	4	E001	P103
E204	Karen	Nowak	F	SENIOR DATA SCIENTIST	AUTOMOTIVE	8	GERMANY	EUROPE	7500	5	E428	P204
E245	Nan	Zhen	M	SENIOR DATA SCIENTIST	RETAIL	6	CHINA	ASIA	6500	2	E583	P109
E260	Roy	Collin	M	SENIOR DATA SCIENTIST	RETAIL	7	INDIA	ASIA	7000	3	E583	NA
E428	Pete	Allan	M	MANAGER	AUTOMOTIVE	14	GERMANY	EUROPE	11000	4	E001	P103
E583	Janet	Hale	F	MANAGER	RETAIL	14	COLOMBIA	SOUTH AMERICA	10000	2	E001	P103
E812	Tracy	Norris	F	MANAGER	RETAIL	13	INDIA	ASIA	8500	4	E001	P103

Action Output:

- 38 15:41:14 CREATE VIEW emp AS SELECT * FROM emp_record_table WHERE SALARY > 6000 Message 0 row(s) affected Duration / Fetch 0.015 sec / 0.000 sec / 0.000 sec
- 39 15:45:08 SELECT * FROM emp LIMIT 0,1000 Message 12 row(s) returned Duration / Fetch 0.000 sec / 0.000 sec / 0.000 sec

Object Info | **Session** | **Query Completed**

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

We have to retrieve the employees with experience of more than ten years using nested query. The query to be executed will be:

```
SELECT x.EMP_ID,x.FIRST_NAME,x.EXP  
FROM emp_record_table x  
WHERE x.EMP_ID IN  
(  
    SELECT y.EMP_ID  
    FROM emp_record_table y  
    WHERE y.EXP > 10  
);
```

Result:

The screenshot shows the MySQL Workbench interface with a query editor and a results grid. The query editor contains the nested SELECT statement provided in the text above. The results grid displays a table with columns EMP_ID, FIRST_NAME, and EXP, showing records for employees with experience greater than 10 years. The bottom section of the interface shows the execution log, indicating the successful execution of the query.

EMP_ID	FIRST_NAME	EXP
E001	Arthur	20
E005	Erik	11
E010	William	12
E012	Patrick	15
E013	Eddy	14
E028	Peter	14
E082	Janet	14
E012	Tracy	13

Execution Log:

- 39 15:45:08 SELECT * FROM emp_record_table LIMIT 0, 1000 12 row(s) returned. 0.000 sec / 0.000 sec
- 40 16:17:12 SELECT x.EMP_ID,x.FIRST_NAME,x.EXP FROM emp_record_table x WHERE x.EMP_ID IN (SELECT y.EMP_ID FROM emp_record_table y WHERE y.EXP > 10) 8 row(s) returned. 0.016 sec / 0.000 sec

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.

Here we have to create a stored procedure to retrieve the details of the employees whose experience is more than three years. The query that is to be executed is:

```
DELIMITER //

CREATE PROCEDURE sp_emp()

BEGIN

    SELECT * FROM emp_record_table

    WHERE EXP > 3;

END; //

DELIMITER ;

CALL sp_emp();
```

Result:

The screenshot shows the MySQL Workbench interface. In the top-left pane, the 'Schemas' tree shows the 'employee' schema selected. The main pane displays the SQL code for creating the stored procedure 'sp_emp'. The bottom pane shows the results of executing the procedure, displaying a table of employee records where 'EXP > 3'. The output window at the bottom shows the execution log, indicating 15 rows returned.

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	E103	P105
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
E012	Dorothy	Wilson	F	SENIOR DATA SCIENTIST	HEALTHCARE	6	CANADA	NORTH AMERICA	9500	3	E003	P103
E057	Dorothy	Wilson	F	SENIOR DATA SCIENTIST	HEALTHCARE	15	USA	NORTH AMERICA	7700	1	E003	P302
E083	Petra	Voltz	M	MANAGER	FINANCIAL	15	USA	NORTH AMERICA	9500	5	E001	P101
E103	Emily	Grove	F	MANAGER	FINANCIAL	24	CANADA	NORTH AMERICA	10500	4	E001	P101
E204	Karen	Novak	F	SENIOR DATA SCIENTIST	AUTOMOTIVE	8	GERMANY	EUROPE	7500	5	E428	P204
E245	Nan	Zhen	M	SENIOR DATA SCIENTIST	RETAIL	6	CHINA	ASIA	6500	2	E583	P109
E260	Roy	Collins	M	SENIOR DATA SCIENTIST	RETAIL	7	INDIA	ASIA	7000	3	E583	NA
E403	Steve	Hoffman	M	ASSOCIATE DATA SCIENTIST	FINANCE	4	USA	NORTH AMERICA	5000	3	E103	P105
E428	Pete	Allen	M	MANAGER	AUTOMOTIVE	14	GERMANY	EUROPE	11000	4	E001	P101
E474	Chad	Johnson	M	ASSOCIATE DATA SCIENTIST	HEALTHCARE	14	CANADA	NORTH AMERICA	6500	3	E003	P101

Output:

```
41 16:25:43 CREATE PROCEDURE sp_emp() BEGIN SELECT * FROM emp_record_table WHERE EXP > 3; END; Message: 0 row(s) affected Duration / Fetch: 0.015 sec / 0.000 sec
42 16:25:43 CALL sp_emp() Message: 15 row(s) returned Duration / Fetch: 0.000 sec / 0.000 sec
```

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'.

We need to check whether the job profile assigned to each employee in the data_science_team matches the organization's set standard by using stored functions in the project_table. In order to check the job profiles, the following steps must be performed

- Declaration of variables such as exist_exp, exist_role, and set_role
- Selecting role and experience from data_science_table for the given employee ID
- Evaluating the existing experience and assigning role as per organization's set standard
- Compare set role to existing role and return the result based on profile matching

The query to be executed will be:

```
DELIMITER //

CREATE FUNCTION fn_prof_match(e_id VARCHAR(4)) RETURNS VARCHAR(50)
DETERMINISTIC

BEGIN

    -- Declaration of variables

    DECLARE exist_exp INT DEFAULT NULL;
    DECLARE exist_role VARCHAR(24) DEFAULT NULL;
    DECLARE set_role VARCHAR(24) DEFAULT NULL;

    -- Selecting experience and role from data_science_team table for the given
    employee id.

    SELECT EXP, ROLE INTO exist_exp, exist_role
    FROM data_science_team
    WHERE EMP_ID = e_id;

    -- Evaluating the existing experience and assigning role as per organization's set
    standard

    IF exist_exp <=2 THEN
        SET set_role = "JUNIOR DATA SCIENTIST";
    ELSEIF exist_exp > 2 AND exist_exp <= 5 THEN
        SET set_role = "ASSOCIATE DATA SCIENTIST";
    ELSEIF exist_exp > 5 AND exist_exp <= 10 THEN
        SET set_role = "SENIOR DATA SCIENTIST";
    ELSEIF exist_exp > 10 AND exist_exp <= 12 THEN
        SET set_role = "SENIOR DATA SCIENTIST";
```

```

        SET set_role = "LEAD DATA SCIENTIST";

        ELSEIF exist_exp > 12 AND exist_exp <= 16 THEN

            SET set_role = "MANAGER";

        END IF;

-- Compare set role to existing role and return the result based on profile matching

        IF exist_role = set_role THEN

            RETURN "Congratulations! Profile matches set standard./";

        ELSE

            RETURN "Sorry! Profile doesn't match set standard./";

        END IF;

    END; //

DELIMITER;

```

The screenshot shows the MySQL Workbench interface with a query editor window. The code in the editor is a stored procedure named sp_employee. It uses an IF-ELSEIF-ELSE structure to determine a role based on experience. It then compares the determined role with the existing role and returns a message accordingly. The code ends with an END; // and a DELIMITER ;.

```

MySQL Workbench - Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help
File Edit View Query Database Server Tools Scripting Help
Navigator Schemas employee pet_shop simplteam sys
Query 1
142 IF exist_exp <=2 THEN
143     SET set_role = "JUNIOR DATA SCIENTIST";
144     ELSEIF exist_exp > 2 AND exist_exp <= 5 THEN
145         SET set_role = "ASSOCIATE DATA SCIENTIST";
146         ELSEIF exist_exp > 5 AND exist_exp <= 10 THEN
147             SET set_role = "SENIOR DATA SCIENTIST";
148             ELSEIF exist_exp > 10 AND exist_exp <= 12 THEN
149                 SET set_role = "LEAD DATA SCIENTIST";
150                 ELSEIF exist_exp > 12 AND exist_exp <= 16 THEN
151                     SET set_role = "MANAGER";
152                 END IF;
153
154 -- Compare set role to existing role and return the result based on profile matching
155         IF exist_role = set_role THEN
156             RETURN "Congratulations! Profile matches set standard./";
157         ELSE
158             RETURN "Sorry! Profile doesn't match set standard./";
159         END IF;
160     END; //
161     DELIMITER ;
162
163
164
165

```

The output pane shows two log entries:

- 42 16:25:43 CALL sp_employee
- 43 18:43:05 CREATE FUNCTION fn_prof_match(role_id VARCHAR(4)) RETURNS VARCHAR(50) DETERMINISTIC BE... 0 row(s) affected

Once the stored function is created as shown in the picture above, use the below query

SHOW FUNCTION STATUS WHERE db = 'employee';

The screenshot shows the MySQL Workbench interface with a query editor window. The code in the editor is:

```
150 ELSEIF exist_exp > 12 AND exist_exp <= 16 THEN
151     SET set_role = "MANAGER";
152 END IF;
153
154 -- Compare set role to existing role and return the result based on profile matching
155 IF exist_role = set_role THEN
156     RETURN "Congratulations! Profile matches set standard.";
157 ELSE
158     RETURN "Sorry! Profile doesn't match set standard.";
159 END IF;
160 END;
161 DELIMITER ;
162 • SHOW FUNCTION STATUS WHERE db = 'employee';
163
164
165
166
```

The Result Grid shows one row of data:

Db	Name	Type	Definer	Modified	Created	Security_type	Comment	character_set_client	collation_connection	Database	Collation
employee	fn_prof_match	FUNCTION	root@localhost	2024-07-31 18:43:05	2024-07-31 18:43:05	DEFINER		utf8mb4	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci	utf8mb4_0900_ai_ci

The Result list shows two entries:

Action	Time	Message	Duration / Fetch
CREATE FUNCTION fn_prof_match(emp_id VARCHAR(4)) RETURNS VARCHAR(50) DETERMINISTIC BE...	43 18:43:05	0 row(s) affected	0.000 sec
SHOW FUNCTION STATUS WHERE db = 'employee'	44 18:47:32	1 row(s) returned	0.000 sec / 0.000 sec

To verify the roles and also to obtain the predefined messages as outcome, execute the below query

Mention the EMP_ID available in the database

SHOW FUNCTION STATUS WHERE db = 'employee';

Result:

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** employee, pet_shop, simplteam, sys
- Query Editor (Query 1):** Contains a stored procedure definition and a SELECT statement:

```
150      ELSEIF exist_exp > 12 AND exist_exp <= 16 THEN
151          SET set_role = "MANAGER";
152      END IF;
153
154      -- Compare set role to existing role and return the result based on profile matching
155      IF exist_role = set_role THEN
156          RETURN "Congratulations! Profile matches set standard.";
157      ELSE
158          RETURN "Sorry! Profile doesn't match set standard.";
159      END IF;
160  END; //
161  DELIMITER ;
162
163 • SHOW FUNCTION STATUS WHERE db = 'employee';
164
165 • SELECT fn_prof_match('E005');
```
- Result Grid (Result 16):** Shows the output of the SELECT statement:

fn_prof_match(E005)
Congratulations! Profile matches set standard.
- Output (Result 16):** Shows the execution log:

44 18:47:32 SHOW FUNCTION STATUS WHERE db = 'employee'	1 row(s) returned
45 18:52:16 SELECT fn_prof_match('E005') LIMIT 0,1000	1 row(s) returned

Mention the EMP_ID not available in the database

```
SELECT fn_prof_match('E007');
```

Result:

The screenshot shows the MySQL Workbench interface with the following details:

- Schemas:** employee, pet_shop, simplteam, sys
- Query Editor (Query 1):** Contains a stored procedure definition and two SELECT statements:

```
152      END IF;
153
154      -- Compare set role to existing role and return the result based on profile matching
155      IF exist_role = set_role THEN
156          RETURN "Congratulations! Profile matches set standard.";
157      ELSE
158          RETURN "Sorry! Profile doesn't match set standard.";
159      END IF;
160  END; //
161  DELIMITER ;
162
163 • SHOW FUNCTION STATUS WHERE db = 'employee';
164
165 • SELECT fn_prof_match('E005'); -- Mention the EMP_ID available in the database
166 • SELECT fn_prof_match('E007'); -- Mention the EMP_ID not available in the database;
```
- Result Grid (Result 17):** Shows the output of the second SELECT statement:

fn_prof_match(E007)
Sorry! Profile doesn't match set standard.
- Output (Result 17):** Shows the execution log:

45 18:52:16 SELECT fn_prof_match('E005') LIMIT 0,1000	1 row(s) returned
46 18:56:30 SELECT fn_prof_match('E007') LIMIT 0,1000	1 row(s) returned

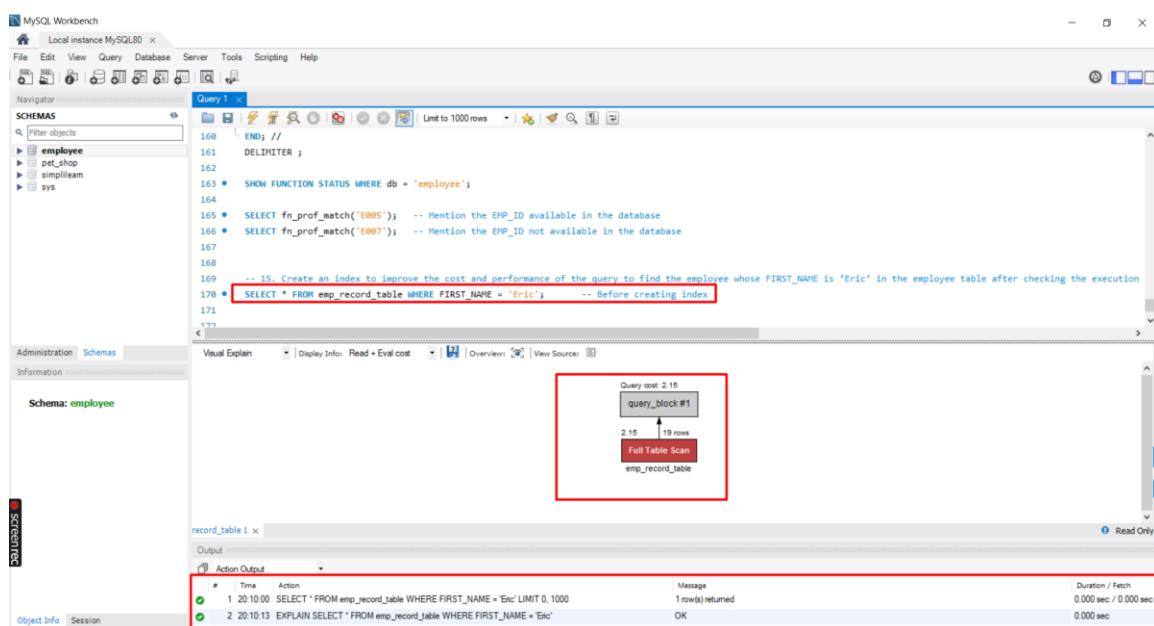
15. Create an index to improve the cost and performance of the query to find the employee whose FIRST_NAME is ‘Eric’ in the employee table after checking the execution plan

We have to 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 by comparing the execution plan before and after creating the index. The query to be executed is:

- Before creating index:

```
SELECT * FROM emp_record_table WHERE FIRST_NAME = 'Eric';
```

Result:



Now we are creating an index using the below query:

```
CREATE INDEX id_emp_fn ON emp_record_table(FIRST_NAME(7));
```

- After creating index:

```
SELECT * FROM emp_record_table WHERE FIRST_NAME = 'Eric';
```

Result:

The screenshot shows the MySQL Workbench interface with a query editor and a visual explain plan.

Query Editor:

```
159      END IF;
160  END //;
161  DELIMITER ;
162
163 *  SHOW FUNCTION STATUS WHERE db = 'employee';
164
165 *  SELECT fn_prof_match('E005'); -- Mention the EMP_ID available in the database
166 *  SELECT fn_prof_match('E007'); -- Mention the EMP_ID not available in the database
167
168
169  -- 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
170 *  SELECT * FROM emp_record_table WHERE FIRST_NAME = 'Eric'; -- Before creating index
171 *  CREATE INDEX id_emp_fn ON emp_record_table(FIRST_NAME); -- creating an index
172 *  SELECT * FROM emp_record_table WHERE FIRST_NAME = 'Eric'; -- After creating index
```

Visual Explain:

Query cost: 0.35
query_block #1
0.35 ↑ 1 row
Non-unique Key Lookup
emp_record_table
id_emp_fn

Output:

#	Time	Action	Message	Duration / Fetch
6	20:16:38	EXPLAIN SELECT * FROM emp_record_table WHERE FIRST_NAME = 'Eric'	OK	0.000 sec
7	20:16:38	EXPLAIN FORMAT=JSON SELECT * FROM emp_record_table WHERE FIRST_NAME = 'Eric'	OK	0.000 sec

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).

We have to calculate the bonus for all the employees based on their ratings and salaries.
The query that is to be executed will be:

SELECT

```
EMP_ID,FIRST_NAME,SALARY,EMP_RATING,(0.05*salary*emp_rating) AS Bonus  
FROM emp_record_table;
```

Result:

The screenshot shows the MySQL Workbench interface with a query editor window titled "Query 1". The query is:`-- 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).
SELECT EMP_ID,FIRST_NAME,SALARY,EMP_RATING,(0.05*salary*emp_rating) AS Bonus
FROM emp_record_table;`

The results are displayed in a table:

EMP_ID	FIRST_NAME	SALARY	EMP_RATING	Bonus
E001	Janet	6500	5	125.00
E002	Eduardo	8500	3	1275.00
E003	Eric	9000	2	900.00
E004	Willians	9000	5	1350.00
E005	Diana	5500	5	1375.00
E006	Dorothy	7700	1	385.00
E007	Patrick	9500	5	2375.00
E008	Emily	10500	4	2100.00
E009	Karen	7500	5	1875.00
E010	Nan	6500	2	650.00
E011	Roy	7000	3	1050.00
E012	Steve	5000	3	750.00
E013	Mark	11000	4	2200.00
E014	Pete	4000	4	800.00
E015	David	4000	4	800.00
E016	Chad	9000	2	900.00
E017	Clare	4300	1	215.00
E018	Janet	10000	2	1000.00
E019	Tracy	8500	4	1700.00
E020	Kathrin	3000	1	150.00
E021	Jenifer	2800	4	560.00

The status bar at the bottom shows the message "19 rows(s) returned".

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

Here we will calculate the average salary distribution for each employee based on the continent and country. The query that is to be executed is:

SELECT

```
EMP_ID,FIRST_NAME,SALARY,COUNTRY,CONTINENT,  
AVG(SALARY) OVER(PARTITION BY COUNTRY)  
AS 'Average Country Based Salary',  
AVG(SALARY) OVER(PARTITION BY CONTINENT)  
AS 'Average Continent Based Salary'
```

FROM emp_record_table;

Result:

The screenshot shows the MySQL Workbench interface with a query editor window. The query is:`178 -- 17. Write a query to calculate the average salary distribution based on the continent and country. Take data from the employee record table.
179
180
181
182
183
184
185`

The results grid displays two columns of average salaries: "Average Country Based Salary" and "Average Continent Based Salary". The data includes rows for employees from various countries and continents, with their respective average salaries. The bottom of the screen shows the MySQL command history with the last two commands highlighted.

EMP_ID	FIRST_NAME	SALARY	COUNTRY	CONTINENT	Average Country Based Salary	Average Continent Based Salary
E245	Nan	6500	CHINA	ASIA	6500.0000	6250.0000
E260	Rey	7000	INDIA	ASIA	6366.6667	6250.0000
E612	Tracy	8500	INDIA	ASIA	6166.6667	6250.0000
E620	Katrina	3000	INDIA	ASIA	6166.6667	6250.0000
E010	William	9000	FRANCE	EUROPE	9000.0000	7950.0000
E204	Karen	7500	GERMANY	EUROPE	7600.0000	7950.0000
E428	Pete	11000	GERMANY	EUROPE	7600.0000	7950.0000
E532	Clare	4300	GERMANY	EUROPE	7600.0000	7950.0000
E052	Dianne	5500	CANADA	NORTH AMERICA	7000.0000	8525.0000
E103	Emily	10500	CANADA	NORTH AMERICA	7000.0000	8525.0000
E070	Chad	8000	CANADA	NORTH AMERICA	7000.0000	8525.0000
E201	Angel	15500	LEBANON	NORTH AMERICA	9440.0000	8525.0000
E005	Eric	8500	USA	NORTH AMERICA	9440.0000	8525.0000
E057	Dorothy	7700	USA	NORTH AMERICA	9440.0000	8525.0000
E083	Patrick	9500	USA	NORTH AMERICA	9440.0000	8525.0000
E403	Steve	5000	USA	NORTH AMERICA	9440.0000	8525.0000
E478	David	4000	COLOMBIA	SOUTH AMERICA	5600.0000	5600.0000
E939	Tammy	10000	PERU	SOUTH AMERICA	5600.0000	5600.0000

Output:
Time Action
8 20:24:18 SELECT EMP_ID,FIRST_NAME,SALARY,EMP_RATING,(0.05*salary)*emp_rating AS Bonus FROM emp... 19 row(s) returned
9 20:37:32 SELECT EMP_ID,FIRST_NAME,SALARY,COUNTRY,CONTINENT, AVG(SALARY) OVER(PARTITION BY COUNTRY) AS 'Average Country Based Salary', AVG(SALARY) OVER(PARTITION BY CONTINENT) AS 'Average Continent Based Salary' FROM emp_record_table; 19 row(s) returned