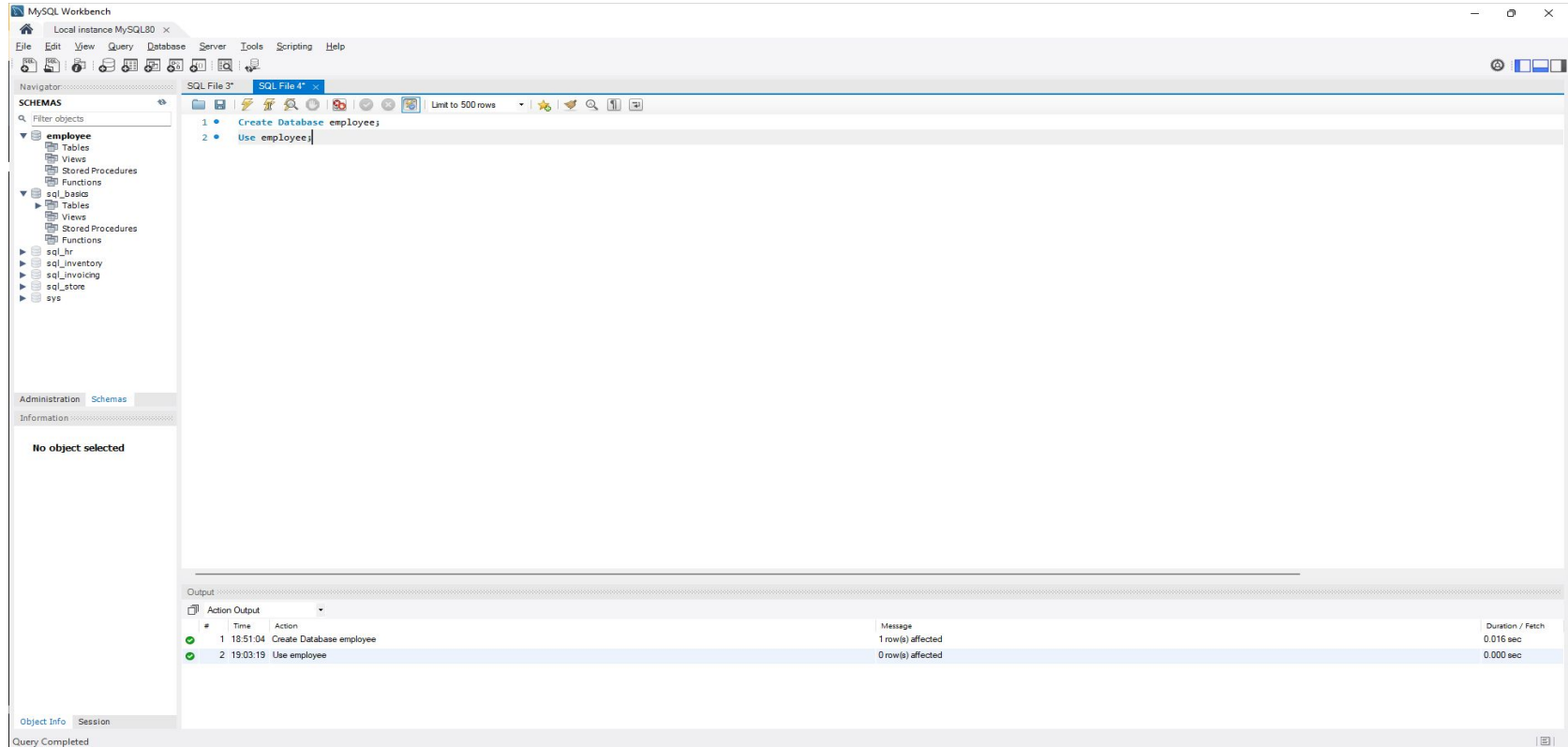


## 1. Create a database named employee.



1. Import **data\_science\_team.csv** **proj\_table.csv** and **emp\_record\_table.csv** into the **employee** database from the given resources.

### Table Data Import

#### Import Results

File C:\Users\Nicky\Downloads\Datasets\_Aug\Datasets\Performance\_Mapping\_Datasets\performance\_mapping\_datasets\emp\_record\_table.csv

Table employee.emp\_record\_table was created

19 records imported

< Back

Finish

### MySQL Workbench

Local instance MySQL80 x

File Edit View Query Database Server Tools Scripting Help

Navigator SQL File 3 SQL File 4 x

SCHEMAS

Filter objects

employee

Tables

emp\_record\_table

Columns

EMP\_ID FIRST\_NAME LAST\_NAME GENDER ROLE DEPT EXP COUNTRY CONTINENT SALARY EMP\_RATING MANAGER\_ID

Indexes Foreign Keys Triggers Stored Procedures Functions Administration Schemas

Table: emp\_record\_table

Columns:

EMP\_ID text

FIRST\_NAME text

LAST\_NAME text

GENDER text

ROLE text

DEPT text

EXP int

COUNTRY text

CONTINENT text

SALARY int

EMP\_RATING int

MANAGER\_ID text

Result Grid Filter Rows: Export Wrap Cell Contents

EMP_ID	FIRST_NAME	LAST_NAME	GENDER	ROLE	DEPT	EXP	COUNTRY	CONTINENT	SALARY	EMP_RATING	MANAGER_ID
E260	Roy	Collins	M	SENIOR DATA SCIENTIST	RETAIL	7	INDIA	ASIA	7000	3	E583
E245	Nian	Zhen	M	SENIOR DATA SCIENTIST	RETAIL	6	CHINA	ASIA	6500	2	E583
E630	Katrina	Allen	F	JUNIOR DATA SCIENTIST	RETAIL	2	INDIA	ASIA	3000	1	E612
E640	Jennifer	Jones	F	JUNIOR DATA SCIENTIST	RETAIL	1	COLOMBIA	SOUTH AMERICA	2800	4	E612
E403	Steve	Hoffman	M	ASSOCIATE DATA SCIENTIST	FINANCE	4	USA	NORTH AMERICA	5000	3	E103
E204	Karen	Novak	F	SENIOR DATA SCIENTIST	AUTOMOTIVE	8	GERMANY	EUROPE	7500	5	E426
E357	Dorothy	Wilson	F	SENIOR DATA SCIENTIST	HEALTHCARE	9	USA	NORTH AMERICA	7700	1	E083
E910	William	Buller	M	LEAD DATA SCIENTIST	AUTOMOTIVE	12	FRANCE	EUROPE	9000	2	E426
E478	David	Smith	M	ASSOCIATE DATA SCIENTIST	RETAIL	3	COLOMBIA	SOUTH AMERICA	4000	4	E583
E605	Eric	Hoffman	M	LEAD DATA SCIENTIST	FINANCE	11	USA	NORTH AMERICA	8500	3	E103
E552	Dianna	Wilson	F	SENIOR DATA SCIENTIST	HEALTHCARE	6	CANADA	NORTH AMERICA	5500	5	E083
E505	Chad	Wilson	M	ASSOCIATE DATA SCIENTIST	HEALTHCARE	5	CANADA	NORTH AMERICA	5000	2	E083
E332	Claire	Brennan	F	ASSOCIATE DATA SCIENTIST	AUTOMOTIVE	3	GERMANY	EUROPE	4300	1	E426
E083	Patrick	Voltz	M	MANAGER	HEALTHCARE	15	USA	NORTH AMERICA	9500	5	E001
E380	Janet	Hale	F	MANAGER	RETAIL	14	COLOMBIA	SOUTH AMERICA	10000	2	E001
E183	Emily	Grove	F	MANAGER	FINANCE	14	CANADA	NORTH AMERICA	10500	4	E001
E612	Tracy	Harris	F	MANAGER	RETAIL	13	INDIA	ASIA	8500	4	E001
E438	Pete	Allen	M	MANAGER	AUTOMOTIVE	14	GERMANY	EUROPE	11000	4	E001
E001	Arthur	Black	M	PRESIDENT	ALL	20	USA	NORTH AMERICA	16500	5	E001

emp\_record\_table2 x

Read Only

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	15:18:01	select * from emp_record_table LIMIT 0, 500	19 row(s) returned	0.000 sec / 0.000 sec

Object Info Session

Query Completed

1. Import **data\_science\_team.csv**, **proj\_table.csv** and **emp\_record\_table.csv** into the **employee** database from the given resources.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' tree with the 'employee' database selected. The 'Tables' list shows 'data\_science\_team'. The 'Columns' list for this table includes: EMP\_ID, FIRST\_NAME, LAST\_NAME, GENDER, ROLE, DEPT, EMP, COUNTRY, and CONTINENT. The 'Table: data\_science\_team' section shows the column details. The 'Result Grid' displays the query result for 'select \* from data\_science\_team;'. The output shows 13 rows of data, including employee details like E260 Roy Collins, E245 Nam Zhen, E620 Karina Allen, etc.

EMP_ID	FIRST_NAME	LAST_NAME	GENDER	ROLE	DEPT	EMP	COUNTRY	CONTINENT
E260	Roy	Collins	M	SENIOR DATA SCIENTIST	RETAIL	7	INDIA	ASIA
E245	Nam	Zhen	M	SENIOR DATA SCIENTIST	RETAIL	6	CHINA	ASIA
E620	Karina	Allen	F	JUNIOR DATA SCIENTIST	RETAIL	2	INDIA	ASIA
E640	Jennifer	Jones	F	JUNIOR DATA SCIENTIST	RETAIL	1	COLOMBIA	SOUTH AMERICA
E403	Steve	Hoffman	M	ASSOCIATE DATA SCIENTIST	FINANCE	4	USA	NORTH AMERICA
E204	Karen	Nowak	F	SENIOR DATA SCIENTIST	AUTOMOTIVE	8	GERMANY	EUROPE
E057	Dorothy	Wilson	F	SENIOR DATA SCIENTIST	HEALTHCARE	9	USA	NORTH AMERICA
E049	William	Buller	M	LEAD DATA SCIENTIST	AUTOMOTIVE	12	FRANCE	EUROPE
E478	David	Smith	M	ASSOCIATE DATA SCIENTIST	RETAIL	3	COLOMBIA	SOUTH AMERICA
E005	Eric	Hoffman	M	LEAD DATA SCIENTIST	FINANCE	11	USA	NORTH AMERICA
E052	Diana	Wilson	F	SENIOR DATA SCIENTIST	HEALTHCARE	6	CANADA	NORTH AMERICA
E055	Chad	Wilson	M	ASSOCIATE DATA SCIENTIST	HEALTHCARE	5	CANADA	NORTH AMERICA
E532	Clare	Brennan	F	ASSOCIATE DATA SCIENTIST	AUTOMOTIVE	3	GERMANY	EUROPE

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'Schemas' tree with the 'employee' database selected. The 'Tables' list shows 'proj\_table'. The 'Columns' list for this table includes: PROJ\_ID, PROJ\_NAME, DOMAIN, START\_DATE, CLOSURE\_DATE, DEV\_QTR, and STATUS. The 'Table: proj\_table' section shows the column details. The 'Result Grid' displays the query result for 'select \* from proj\_table;'. The output shows 10 rows of data, including project details like P103 Drug Discovery, P105 Fraud Detection, P208 Algorithmic Trading, etc.

PROJ_ID	PROJ_NAME	DOMAIN	START_DATE	CLOSURE_DATE	DEV_QTR	STATUS
P103	Drug Discovery	HEALTHCARE	04-06-2021	6/20/2021	Q1	DONE
P105	Fraud Detection	FINANCE	04-11-2021	6/25/2021	Q1	DONE
P208	Algorithmic Trading	FINANCE	01/16/2022	3/27/2022	Q4	YTS
P109	Market Basket Analysis	RETAIL	04-12-2021	6/30/2021	Q1	DELAIED
P204	Supply Chain Management	AUTOMOTIVE	07/15/2021	9/28/2021	Q2	WIP
P406	Customer Sentiment Analysis	RETAIL	07-09-2021	9/24/2021	Q2	WIP
P302	Early Detection of Lung Cancer	HEALTHCARE	10-08-2021	12/18/2021	Q3	YTS
P201	Self Driving Cars	AUTOMOTIVE	01-12-2022	3/30/2022	Q4	YTS

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

The screenshot displays the MySQL Workbench interface with an EER Diagram for the 'employee' database. The diagram is set on a grid background and contains three tables:

- proj\_table**
  - PROJ\_ID TEXT
  - PROJ\_NAME TEXT
  - DOMAIN TEXT
  - START\_DATE TEXT
  - CLOSURE\_DATE TEXT
  - DEV\_QTR TEXT
  - STATUS TEXT
- emp\_record\_table**
  - EMP\_ID TEXT
  - FIRST\_NAME TEXT
  - LAST\_NAME TEXT
  - GENDER TEXT
  - ROLE TEXT
  - DEPT TEXT
  - EXP INT
  - COUNTRY TEXT
  - CONTINENT TEXT
  - SALARY INT
  - EMP\_RATING INT
  - MANAGER ID TEXT
- data\_science\_team**
  - EMP\_ID TEXT
  - FIRST\_NAME TEXT
  - LAST\_NAME TEXT
  - GENDER TEXT
  - ROLE TEXT
  - DEPT TEXT
  - EXP INT
  - COUNTRY TEXT
  - CONTINENT TEXT

The interface includes a 'Catalog Tree' on the left showing the database structure, a 'Diagram' tab, and a 'Modeling Additions' panel on the right with tables like 'timestamps', 'user', and 'category'. The status bar at the bottom indicates 'Ready'.

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.

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' pane displays a tree view of the database structure, including tables like 'emp\_record\_table'. The main query editor contains the following SQL query:

```
1 • select EMP_ID, FIRST_NAME, LAST_NAME, GENDER,DEPT from emp_record_table;
2
```

The 'Result Grid' shows the results of the query, displaying columns EMP\_ID, FIRST\_NAME, LAST\_NAME, GENDER, and DEPT. The data is as follows:

EMP_ID	FIRST_NAME	LAST_NAME	GENDER	DEPT
E260	Roy	Collins	M	RETAIL
E245	Nian	Zhen	M	RETAIL
E620	Katrina	Allen	F	RETAIL
E640	Jennifer	Jhones	F	RETAIL
E403	Steve	Hoffman	M	FINANCE
E204	Karene	Nowak	F	AUTOMOTIVE
E057	Dorothy	Wilson	F	HEALTHCARE
E010	William	Butler	M	AUTOMOTIVE
E478	David	Smith	M	RETAIL
E005	Eric	Hoffman	M	FINANCE
E052	Dianna	Wilson	F	HEALTHCARE
E505	Chad	Wilson	M	HEALTHCARE
E532	Claire	Brennan	F	AUTOMOTIVE
E083	Patrick	Voltz	M	HEALTHCARE
E583	Janet	Hale	F	RETAIL
E103	Emily	Grove	F	FINANCE
E612	Tracy	Norris	F	RETAIL
E428	Pete	Allen	M	AUTOMOTIVE
E001	Arthur	Black	M	ALL

The 'Output' pane at the bottom shows the execution details:

#	Time	Action	Message	Duration / Fetch
1	19:38:59	select EMP_ID, FIRST_NAME, LAST_NAME, GENDER,DEPT from emp_record_table LIMIT 0, 500	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

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following query:

```
1 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 between 2 and 4 ;
2
```

The Results Grid displays 19 rows of data:

EMP_ID	FIRST_NAME	LAST_NAME	GENDER	DEPT	EMP_RATING
E260	Roy	Collins	M	RETAIL	3
E245	Nian	Zhen	M	RETAIL	2
E620	Katrina	Allen	F	RETAIL	1
E640	Jenifer	Jhones	F	RETAIL	4
E403	Steve	Hoffman	M	FINANCE	3
E204	Karene	Nowak	F	AUTOMOTIVE	5
E057	Dorothy	Wilson	F	HEALTHCARE	1
E010	William	Butler	M	AUTOMOTIVE	2
E478	David	Smith	M	RETAIL	4
E005	Eric	Hoffman	M	FINANCE	3
E052	Dianna	Wilson	F	HEALTHCARE	5
E505	Chad	Wilson	M	HEALTHCARE	2
E532	Claire	Brennan	F	AUTOMOTIVE	1
E083	Patrick	Voltz	M	HEALTHCARE	5
E583	Janet	Hale	F	RETAIL	2
E103	Emily	Grove	F	FINANCE	4
E612	Tracy	Norris	F	RETAIL	4
E428	Pete	Allen	M	AUTOMOTIVE	4
E001	Arthur	Black	M	ALL	5

The Output pane at the bottom shows the execution details:

```
1 19:45:40 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 between 2 and 4 ; 19 row(s) returned
```

Duration / Fetch: 0.000 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.

The screenshot shows the MySQL Workbench interface. The SQL Editor contains the following query:

```
1 select concat(FIRST_NAME, ' ', LAST_NAME) as NAME from emp_record_table where DEPT='FINANCE';
2
3
```

The Results Grid shows the output of the query:

NAME
Steve Hoffman
Eric Hoffman
Emily Grove

The bottom panel shows the Output tab with the following message:

```
1 19:55:45 select concat(FIRST_NAME, ' ', LAST_NAME) as NAME from emp_record_table where DEPT='FINANCE' LIMIT 0, 500
3 row(s) returned
```

The bottom right corner shows the Duration / Fetch: 0.000 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).

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with the 'employee' database selected. The 'Tables' list includes 'data\_science\_team', 'emp\_record\_table', and 'prod\_table'. The 'Views' list includes 'Stored Procedures', 'Functions', and 'sys'. The 'Information' tab is active, showing the 'Table: emp\_record\_table' with columns: EMP\_ID (text), FIRST\_NAME (text), LAST\_NAME (text), GENDER (text), ROLE (text), DEPT (text), EXP (int), COUNTRY (text), CONTINENT (text), SALARY (int), EMP\_RATING (int), and MANAGER\_ID (text).

The main editor window shows a SQL query in 'SQL File 8\*':

```
1 SELECT e1.manager_id,
2    count(*)
3 FROM emp_record_table e1,
4    emp_record_table e2
5 WHERE e1.manager_id = e2.emp_id
6 GROUP BY e1.manager_id
7 ORDER BY e1.manager_id ASC;
```

The 'Result Grid' shows the following data:

manager_id	count(*)
E001	6
E083	3
E103	2
E428	3
E583	3
E612	2

The bottom status bar indicates 'Query Completed'.



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.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with the 'employee' database selected. The 'Tables' list under 'employee' includes 'data\_science\_team', 'emp\_record\_table', and 'prod\_table'. The 'emp\_record\_table' is highlighted. The bottom-left pane shows the 'Table: emp\_record\_table' structure with columns: EMP\_ID (text), FIRST\_NAME (text), LAST\_NAME (text), GENDER (text), ROLE (text), DEPT (text), EXP (int), COUNTRY (text), CONTINENT (text), SALARY (int), EMP\_RATING (int), and MANAGER\_ID (text).

The main editor window shows a SQL query in 'SQL File 8\*':

```
1 • select emp_id from emp_record_table where dept='FINANCE';
2 union
3 select emp_id from emp_record_table where dept='HEALTHCARE';
```

The 'Result Grid' pane displays the results of the query, showing a list of employee IDs (emp\_id) for the 'FINANCE' and 'HEALTHCARE' departments. The results are:

emp_id
E403
E005
E103
E057
E052
E505
E083

The bottom status bar indicates 'Query Completed'.

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.

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
1 • select EMP_ID, FIRST_NAME, LAST_NAME, ROLE, DEPT, EMP_RATING, max(EMP_RATING) over(partition by dept) from emp_record_table ;
```

The Results window displays the output of the query in a grid format. The columns are EMP\_ID, FIRST\_NAME, LAST\_NAME, ROLE, DEPT, EMP\_RATING, and max(EMP\_RATING) over(partition by dept). The results show 17 rows of employee data, grouped by department.

EMP_ID	FIRST_NAME	LAST_NAME	ROLE	DEPT	EMP_RATING	max(EMP_RATING) over(partition by dept)
E001	Arthur	Black	PRESIDENT	ALL	5	5
E204	Karene	Nowak	SENIOR DATA SCIENTIST	AUTOMOTIVE	5	5
E010	William	Butler	LEAD DATA SCIENTIST	AUTOMOTIVE	2	5
E532	Clare	Brennan	ASSOCIATE DATA SCIENTIST	AUTOMOTIVE	1	5
E428	Pete	Allen	MANAGER	AUTOMOTIVE	4	5
E403	Steve	Hoffman	ASSOCIATE DATA SCIENTIST	FINANCE	3	4
E005	Eric	Hoffman	LEAD DATA SCIENTIST	FINANCE	3	4
E103	Emily	Grove	MANAGER	FINANCE	4	4
E057	Dorothy	Wilson	SENIOR DATA SCIENTIST	HEALTHCARE	1	5
E052	Dianna	Wilson	SENIOR DATA SCIENTIST	HEALTHCARE	5	5
E505	Chad	Wilson	ASSOCIATE DATA SCIENTIST	HEALTHCARE	2	5
E083	Patrick	Volz	MANAGER	HEALTHCARE	5	5
E260	Roy	Collins	SENIOR DATA SCIENTIST	RETAIL	3	4
E245	Nian	Zhen	SENIOR DATA SCIENTIST	RETAIL	2	4
E520	Katrina	Allen	JUNIOR DATA SCIENTIST	RETAIL	1	4
E640	Jenifer	Jhones	JUNIOR DATA SCIENTIST	RETAIL	4	4
E478	David	Smith	ASSOCIATE DATA SCIENTIST	RETAIL	4	4
E583	Janet	Hale	MANAGER	RETAIL	2	4
E612	Tracy	Norris	MANAGER	RETAIL	4	4

The bottom of the interface shows the execution details: 1 row(s) returned, 0.000 sec / 0.000 sec.

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.

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
1 select min(salary),max(salary) from emp_record_table group by role;
```

The query is executed, and the results are displayed in the Result Grid. The results show the minimum and maximum salary for each role.

min(salary)	max(salary)
5500	7700
2800	3000
4000	5000
8500	9000
8500	11000
16500	16500

The bottom panel shows the Output tab with the following message:

```
1 10:43:29 select min(salary),max(salary) from emp_record_table group by role LIMIT 0.500
6 row(s) returned
```

The Duration / Fetch is 0.000 sec / 0.000 sec.

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

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with the 'employee' database selected. The 'Tables' section is expanded, showing 'emp\_record\_table'. The 'Columns' section for 'emp\_record\_table' lists: EMP\_ID (text), FIRST\_NAME (text), LAST\_NAME (text), GENDER (text), ROLE (text), DEPT (text), EXP (int), COUNTRY (text), CONTINENT (text), SALARY (int), EMP\_RATING (int), and MANAGER\_ID (text).

The main query editor shows the following SQL query:

```
1 select emp_id,first_name,role,exp,rank() over(order by exp) emp_rank from emp_record_table;
2
```

The 'Result Grid' displays the query results with columns: emp\_id, first\_name, role, exp, and emp\_rank. The results are as follows:

emp_id	first_name	role	exp	emp_rank
E640	Jennifer	JUNIOR DATA SCIENTIST	1	1
E620	Kathrina	JUNIOR DATA SCIENTIST	2	2
E478	David	ASSOCIATE DATA SCIENTIST	3	3
E532	Claire	ASSOCIATE DATA SCIENTIST	3	3
E403	Steve	ASSOCIATE DATA SCIENTIST	4	5
E505	Chad	ASSOCIATE DATA SCIENTIST	5	6
E245	Nan	SENIOR DATA SCIENTIST	6	7
E052	Danna	SENIOR DATA SCIENTIST	6	7
E260	Roy	SENIOR DATA SCIENTIST	7	9
E204	Karene	SENIOR DATA SCIENTIST	8	10
E057	Dorothy	SENIOR DATA SCIENTIST	9	11
E005	Eric	LEAD DATA SCIENTIST	11	12
E010	William	LEAD DATA SCIENTIST	12	13
E612	Tracy	MANAGER	13	14
E583	Janet	MANAGER	14	15
E403	Emily	MANAGER	14	15
E428	Pete	MANAGER	14	15
E083	Patrick	MANAGER	15	18
E001	Arthur	PRESIDENT	20	19

The 'Output' section at the bottom shows the execution details:

#	Time	Action	Message	Duration / Fetch
1	10.45.26	select emp_id,first_name,role,exp,rank() over(order by exp) emp_rank from emp_record_table	19 row(s) returned	0.000 sec / 0.000 sec

The status bar at the bottom indicates 'Query Completed'.

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.

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
1 Create view emp_country as select * from emp_record_table where salary > 6000;
```

The left sidebar shows the 'SCHEMAS' tree with the 'employee' database selected. Under 'Views', the 'emp\_country' view is highlighted. Below the tree, the 'View: emp\_country' details are shown:

**Columns:**

Column Name	Data Type
EMP_ID	text
FIRST_NAME	text
LAST_NAME	text
GENDER	text
ROLE	text
DEPT	text
EXP	int
COUNTRY	text
CONTINENT	text
SALARY	int
EMP_RATING	int
MANAGER_ID	text

The bottom panel shows the 'Output' tab with the following message:

#	Time	Action	Message	Duration / Fetch
1	10:52:30	Create view emp_country as select * from emp_record_table where salary > 6000	0 row(s) affected	0.016 sec

The status bar at the bottom indicates 'Query Completed'.

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

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
1 • select EMP_ID,EXP from emp_record_table where emp_id in (select EMP_ID from emp_record_table where exp>10) ;
```

The Results Grid displays the following data:

EMP_ID	EXP
E010	12
E005	11
E083	15
E583	14
E103	14
E612	13
E428	14
E001	20

The left sidebar shows the Schemas pane with the database structure. The bottom pane shows the Output tab with the following message:

```
emp_record_table 23 x  
Output  
Action Output  
# Time Action Message Duration / Fetch  
1 10:50:42 select EMP_ID,EXP from emp_record_table where emp_id in (select EMP_ID from emp_record_table where exp>10) LIMIT 0, 500 8 row(s) returned 0.000 sec / 0.000 sec
```

Query Completed

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.

The screenshot displays the MySQL Workbench interface. On the left, the 'Navigator' pane shows the 'employee' schema with its various components. The main editor window contains the following SQL code:

```
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.*/
2 DELIMITER //
3
4 Create Procedure Emp_Detail_Exp()
5 Begin
6     Select * from emp_record_table where exp> 3;
7 End //
8
9 DELIMITER ;
```

At the bottom, the 'Output' pane shows the execution results:

#	Time	Action	Message	Duration / Fetch
1	08:51:53	Create Procedure Emp_Detail_Exp() Begin Select * from emp_record_table where exp> 3; End	0 row(s) affected	0.031 sec

The status bar at the bottom indicates 'Query Completed'.

### 13. Call Procedure.

The screenshot displays the MySQL Workbench interface. The left sidebar shows the 'SCHEMAS' tree with the 'employee' schema selected. The main editor window contains the following SQL code:

```
2 DELIMITER //
3
4 • Create Procedure Emp_Detail_Exp()
5 Begin
6   Select * from emp_record_table where exp> 3;
7 End //
8
9 DELIMITER ;
10
11 • Call Emp_Detail_Exp();
```

Below the code editor, the 'Result Grid' shows the output of the procedure call. The grid has 12 columns: EMP\_ID, FIRST\_NAME, LAST\_NAME, GENDER, ROLE, DEPT, EXP, COUNTRY, CONTINENT, SALARY, EMP\_RATING, and MANAGER\_ID. The results are as follows:

EMP_ID	FIRST_NAME	LAST_NAME	GENDER	ROLE	DEPT	EXP	COUNTRY	CONTINENT	SALARY	EMP_RATING	MANAGER_ID
E260	Roy	Collins	M	SENIOR DATA SCIENTIST	RETAIL	7	INDIA	ASIA	7000	3	E583
E245	Nan	Zhen	M	SENIOR DATA SCIENTIST	RETAIL	6	CHINA	ASIA	6500	2	E583
E403	Steve	Hoffman	M	ASSOCIATE DATA SCIENTIST	FINANCE	4	USA	NORTH AMERICA	5000	3	E103
E204	Karene	Nowak	F	SENIOR DATA SCIENTIST	AUTOMOTIVE	8	GERMANY	EUROPE	7500	5	E428
E057	Dorothy	Wilson	F	SENIOR DATA SCIENTIST	HEALTHCARE	9	USA	NORTH AMERICA	7700	1	E083
E030	Willem	Butler	M	LEAD DATA SCIENTIST	AUTOMOTIVE	12	FRANCE	EUROPE	9000	2	E428
E005	Eric	Hoffman	M	LEAD DATA SCIENTIST	FINANCE	11	USA	NORTH AMERICA	8500	3	E103
E052	Danna	Wilson	F	SENIOR DATA SCIENTIST	HEALTHCARE	6	CANADA	NORTH AMERICA	5500	5	E083
E505	Chad	Wilson	M	ASSOCIATE DATA SCIENTIST	HEALTHCARE	5	CANADA	NORTH AMERICA	5000	2	E083
E083	Patrick	Voltz	M	MANAGER	HEALTHCARE	15	USA	NORTH AMERICA	9500	5	E001
E583	Janet	Hale	F	MANAGER	RETAIL	14	COLOMBIA	SOUTH AMERICA	10000	2	E001
E103	Emily	Grove	F	MANAGER	FINANCE	14	CANADA	NORTH AMERICA	10500	4	E001
E612	Tracy	Norris	F	MANAGER	RETAIL	13	INDIA	ASIA	8500	4	E001
E428	Pete	Allen	M	MANAGER	AUTOMOTIVE	14	GERMANY	EUROPE	11000	4	E001
E001	Arthur	Black	M	PRESIDENT	ALL	20	USA	NORTH AMERICA	16500	5	E001

At the bottom, the 'Output' tab shows the execution log:

#	Time	Action	Message	Duration / Fetch
1	08:51:53	Create Procedure Emp_Detail_Exp() Begin Select * from emp_record_table where exp> 3; End	0 row(s) affected	0.031 sec
2	08:53:57	Call Emp_Detail_Exp()	15 row(s) returned	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'.

The screenshot displays the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows a database named 'employee' with various tables and functions. The 'Functions' section is expanded, showing a function named 'f() Org\_Exp\_Std'. The main editor window shows the SQL code for creating this function. The code defines a function 'Org\_Exp\_Std' that takes 'exp\_year' as an integer and returns a 'VARCHAR(25)'. It uses a series of 'IF' and 'ELSEIF' statements to assign job profiles based on experience ranges: 'JUNIOR DATA SCIENTIST' for 0-2 years, 'ASSOCIATE DATA SCIENTIST' for 2-5 years, 'SENIOR DATA SCIENTIST' for 5-10 years, 'LEAD DATA SCIENTIST' for 10-12 years, and 'MANAGER' for 12-16 years. The function is deterministic and returns the role found from the 'emp\_record\_table'. The output pane at the bottom shows the successful execution of the function creation query.

```
1 DELIMITER //
2 * Create Function Org_Exp_Std( exp_year int)
3 Returns Varchar(25)
4 Deterministic
5 Begin
6     Declare role_found varchar(25);
7     If exp_year <=2 Then
8         Select distinct role into role_found from emp_record_table where exp =exp_year;
9     Elseif (exp_year >2 And exp_year<=5) Then
10        Select distinct role into role_found from emp_record_table where exp =exp_year;
11    Elseif (exp_year >5 And exp_year<=10) Then
12        Select distinct role into role_found from emp_record_table where exp =exp_year;
13    Elseif (exp_year >10 And exp_year<=12) Then
14        Select distinct role into role_found from emp_record_table where exp =exp_year;
15    Elseif (exp_year >12 And exp_year<=16) Then
16        Select distinct role into role_found from emp_record_table where exp =exp_year;
17    End If;
18    Return (role_found);
19 End //
20
21 DELIMITER $$
22
23
```

Output:

#	Time	Action	Message	Duration / Fetch
1	09:55:44	Create Function Org_Exp_Std(exp_year int) Returns Varchar(25) Deterministic Begin	Declare role_found varchar(25); If exp_year <=2 Then Select d... 0 row(s) affected	0.016 sec

Query Completed

## 14. Call Function.

The screenshot displays the MySQL Workbench interface. On the left, the 'SCHEMAS' pane shows a tree view of the database structure, including tables, views, and functions. The 'employee' schema is expanded, showing tables like 'data\_science\_team', 'emp\_record\_table', and 'org\_exp\_std'. The 'Functions' folder is also visible. The main editor window shows a SQL script defining a function 'Org\_Exp\_Std' and then calling it. The script is as follows:

```
3 Returns Varchar(25)
4 Deterministic
5 Begin
6   Declare role_found varchar(25);
7   If exp_year <=2 Then
8     Select distinct role into role_found from emp_record_table where exp =exp_year;
9   Elseif (exp_year >2 And exp_year<=5) Then
10    Select distinct role into role_found from emp_record_table where exp =exp_year;
11  Elseif (exp_year >5 And exp_year<=10) Then
12    Select distinct role into role_found from emp_record_table where exp =exp_year;
13  Elseif (exp_year >10 And exp_year<=13) Then
14    Select distinct role into role_found from emp_record_table where exp =exp_year;
15  Elseif (exp_year >13 And exp_year<=16) Then
16    Select distinct role into role_found from emp_record_table where exp =exp_year;
17  End If;
18  Return (role_found);
19 End //
20
21 DELIMITER $$
22
23 • select Org_Exp_Std(exp),exp from emp_record_table;
```

Below the script, the 'Result Grid' shows the output of the function call. The columns are 'Org\_Exp\_Std(exp)' and 'exp'. The data is as follows:

Org_Exp_Std(exp)	exp
SENIOR DATA SCIENTIST	7
SENIOR DATA SCIENTIST	6
JUNIOR DATA SCIENTIST	2
JUNIOR DATA SCIENTIST	1
ASSOCIATE DATA SCIENTIST	4
SENIOR DATA SCIENTIST	8
SENIOR DATA SCIENTIST	9
LEAD DATA SCIENTIST	12
ASSOCIATE DATA SCIENTIST	3
LEAD DATA SCIENTIST	11
SENIOR DATA SCIENTIST	6

The 'Output' pane at the bottom shows the execution log. It includes the following entries:

Time	Action	Message	Duration / Patch
1 09:55:44	Create Function Org_Exp_Std(exp int) Returns Varchar(25) Deterministic Begin	Declare role_found varchar(25); If exp_year <=2 Then Sele...	0 row(s) affected 0.016 sec
2 09:58:08	Call Org_Exp_Std(4)	Error Code: 1305. PROCEDURE employee.Org_Exp_Std does not exist	0.000 sec
3 09:59:59	select Org_Exp_Std(exp) from emp_record_table LIMIT 0, 500	19 row(s) returned	0.016 sec / 0.000 sec

The 'Query Completed' message is shown at the bottom left.

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.

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'SCHEMAS' tree with the 'employee' database selected. The 'emp\_record\_table' is highlighted. The main editor window shows a SQL query: `Create Index F_Name_INDEX on emp_record_table(First_name(30));`. The bottom status bar indicates 'Query Completed'. The bottom right pane shows the 'Output' tab with a table of results.

#	Time	Action	Message	Duration / Fetch
1	11:21:40	Create Index F_Name_INDEX on emp_record_table(First_name(30))	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.110 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).

The screenshot shows the MySQL Workbench interface. The SQL editor contains the following query:

```
1 /*5% of salary * employee rating*/
2
3 select salary, emp_rating, round(((5 * Salary * Emp_rating)/100),2) as Bonus from emp_record_table;
```

The query results are displayed in a table with the following data:

salary	emp_rating	Bonus
7000	3	1050.00
6500	2	650.00
3000	1	150.00
2800	4	560.00
5000	3	750.00
7500	5	1875.00
7700	1	385.00
9000	2	900.00
4000	4	800.00
8500	3	1275.00
5500	5	1375.00
5000	2	500.00
4300	1	215.00
9500	5	2375.00
10000	2	1000.00
10500	4	2100.00
8500	4	1700.00
11000	4	2200.00
16500	5	4125.00

The bottom panel shows the execution output:

#	Time	Action	Message	Duration / Fetch
1	11:34:51	select salary, emp_rating, round(((5 * Salary * Emp_rating)/100),2) as Bonus from emp_record_table LIMIT 0, 500	19 row(s) returned	0.000 sec / 0.000 sec

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

The screenshot shows the MySQL Workbench interface. In the 'Query' tab, the following SQL query is entered:

```
1 • select salary, round(avg(salary),2) as avg_salary, country, continent  
2 | from emp_record_table group by CONTINENT, COUNTRY;
```

The 'Result Grid' shows the output of the query:

salary	avg_salary	country	continent
7000	6166.67	INDIA	ASIA
6500	6500.00	CHINA	ASIA
2800	5600.00	COLOMBIA	SOUTH AMERICA
5000	9440.00	USA	NORTH AMERICA
7500	7600.00	GERMANY	EUROPE
9000	9000.00	FRANCE	EUROPE
5500	7000.00	CANADA	NORTH AMERICA

The left sidebar shows the 'SCHEMAS' panel with the 'employee' database selected. The 'emp\_record\_table' table is highlighted, showing its columns: EMP\_ID, FIRST\_NAME, LAST\_NAME, GENDER, ROLE, DEPT, EXP, COUNTRY, CONTINENT, SALARY, EMP\_RATING, and MANAGER\_ID. The 'Table: emp\_record\_table' section in the bottom left provides a detailed view of these columns and their data types.