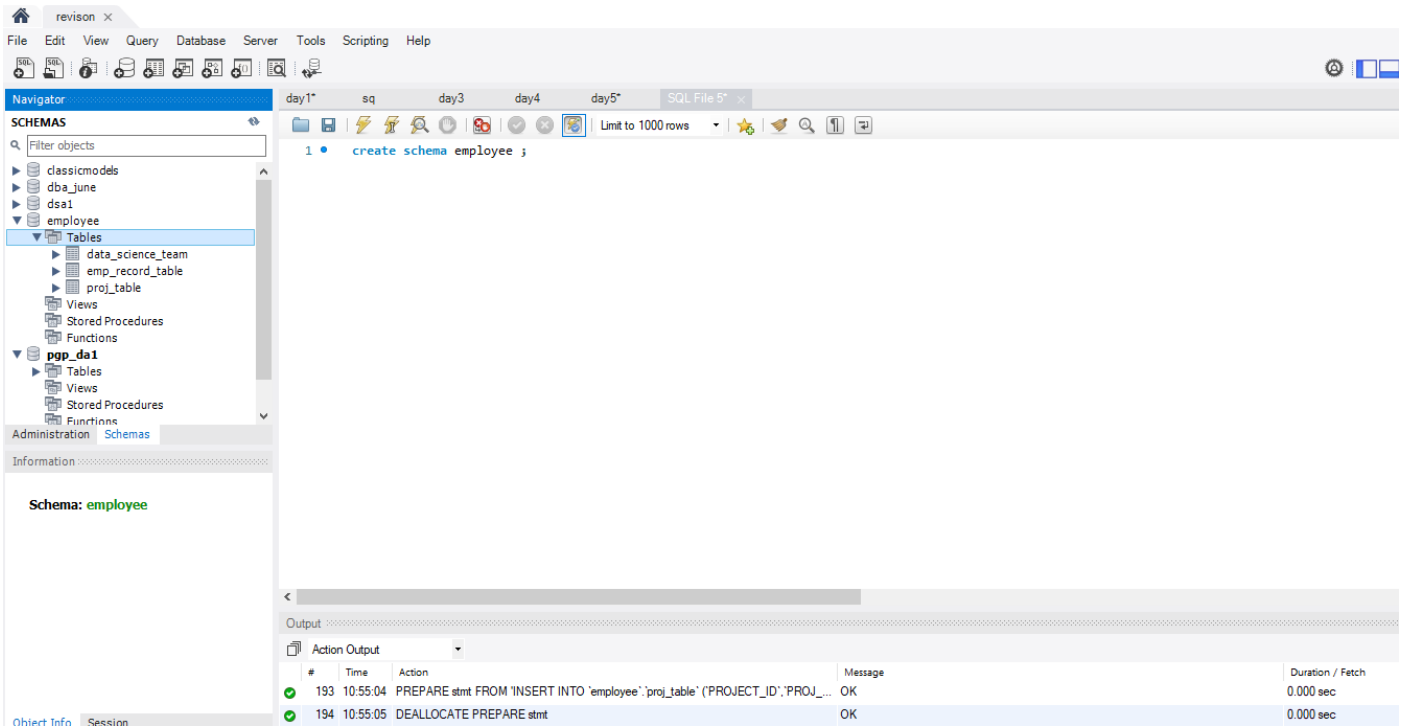
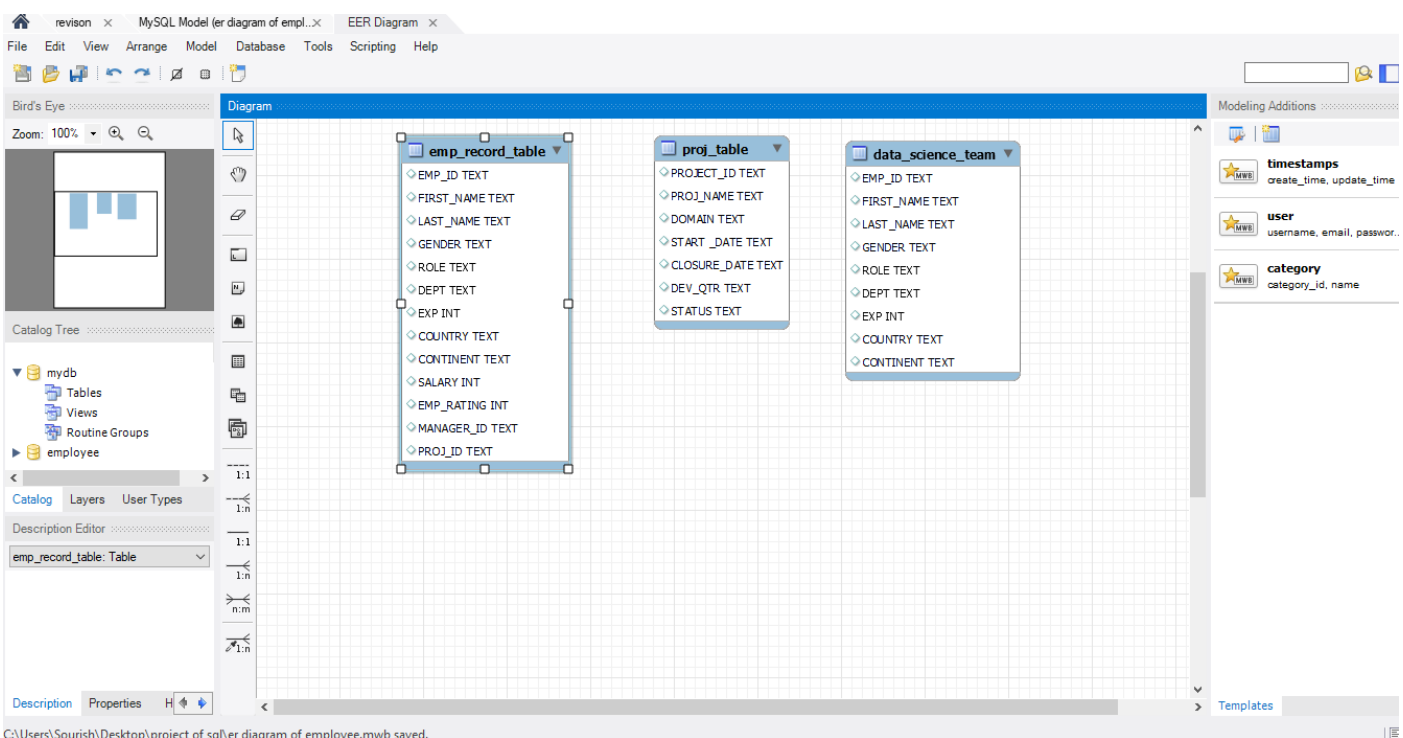


# ScienceQtech Employee Performance Mapping

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.



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



C:\Users\Sourish\Desktop\project of sql\er diagram of employee.mwb saved.

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.

Ans-

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

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

```
1 create schema employee ;  
2 use employee ;  
3  
4 SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT AS DEPARTMENT  
5 FROM emp_record_table;
```

The result grid displays the following data:

EMP_ID	FIRST_NAME	LAST_NAME	GENDER	DEPARTMENT
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

The output section shows the following message:

```
1 11:24:59 SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT AS DEPARTMENT ... 19 row(s) returned
```

Duration / Fetch: 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

=>

### -- EMP\_RATING less than two

The screenshot shows the SQL Developer interface with a query window titled 'SQL File 5\*'. The query is as follows:

```
5 FROM emp_record_table;
6
7
8
9 -- EMP_RATING less than two
10 SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT, EMP_RATING
11 FROM emp_record_table
12 WHERE EMP_RATING < 2;
13
14
15
```

The 'Result Grid' displays the following data:

EMP_ID	FIRST_NAME	LAST_NAME	GENDER	DEPT	EMP_RATING
E057	Dorothy	Wilson	F	HEALTHCARE	1
E532	Claire	Brennan	F	AUTOMOTIVE	1
E620	Katrina	Allen	F	RETAIL	1

The 'Output' pane shows the execution message: 'SELECT EMP\_ID, FIRST\_NAME, LAST\_NAME, GENDER, DEPT, EMP\_RATING FRO... 3 row(s) returned'.

### -- EMP\_RATING greater than four

The screenshot shows the SQL Developer interface with a query window titled 'SQL File 5\*'. The query is as follows:

```
11 FROM emp_record_table
12 WHERE EMP_RATING < 2;
13
14 -- EMP_RATING greater than four
15 SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT, EMP_RATING
16 FROM emp_record_table
17 WHERE EMP_RATING > 4;
18
19
20
```

The 'Result Grid' displays the following data:

EMP_ID	FIRST_NAME	LAST_NAME	GENDER	DEPT	EMP_RATING
E001	Arthur	Black	M	ALL	5
E052	Dianna	Wilson	F	HEALTHCARE	5
E083	Patrick	Voltz	M	HEALTHCARE	5
E204	Karene	Nowak	F	AUTOMOTIVE	5

The 'Output' pane shows two execution messages:

- 1 11:31:42 SELECT EMP\_ID, FIRST\_NAME, LAST\_NAME, GENDER, DEPT, EMP\_RATING FRO... 3 row(s) returned 0.000 sec / 0.000 sec
- 2 11:34:10 SELECT EMP\_ID, FIRST\_NAME, LAST\_NAME, GENDER, DEPT, EMP\_RATING FRO... 4 row(s) returned 0.000 sec / 0.000 sec

## -- EMP\_RATING between two and four

The screenshot shows the SQL Developer interface with a query window titled "SQL File 5". The query is as follows:

```
-- EMP_RATING between two and four
SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT, EMP_RATING
FROM emp_record_table
WHERE EMP_RATING >= 2 AND EMP_RATING <= 4;
```

The "Result Grid" displays the following data:

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	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
E583	Janet	Hale	F	RETAIL	2
E612	Tracy	Norris	F	RETAIL	4
E640	Jenifer	Jhones	F	RETAIL	4

The "Output" window shows the execution results:

#	Time	Action	Message	Duration / Fetch
2	11:34:10	SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT, EMP_RATING FR...	4 row(s) returned	0.000 sec / 0.000 sec
3	11:36:24	SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT, EMP_RATING FR...	12 row(s) returned	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 SQL Developer interface with a query window titled "sql employee project". The query is as follows:

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

The "Result Grid" displays the following data:

NAME
Eric Hoffman
Emily Grove
Steve Hoffman

The "Output" window shows the execution results:

#	Time	Action	Message	Duration / Fetch
3	11:36:24	SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT, EMP_RATING FR...	12 row(s) returned	0.000 sec / 0.000 sec
4	11:40:14	SELECT CONCAT(FIRST_NAME, ' ', LAST_NAME) AS NAME FROM emp_record_ta...	3 row(s) returned	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 SQL Developer interface with a query window open. The query is as follows:

```

29
30
31 • SELECT e.EMP_ID, e.FIRST_NAME, e.LAST_NAME, e.GENDER, e.DEPT AS DEPARTMENT,
32       COUNT(*) AS NUM_REPORTERS
33 FROM emp_record_table e
34 JOIN emp_record_table r ON e.EMP_ID = r.MANAGER_ID
35 GROUP BY e.EMP_ID, e.FIRST_NAME, e.LAST_NAME, e.GENDER, e.DEPT
36 HAVING COUNT(*) > 0;
37
38

```

The result grid shows the following data:

EMP_ID	FIRST_NAME	LAST_NAME	GENDER	DEPARTMENT	NUM_REPORTERS
E103	Emily	Grove	F	FINANCE	2
E428	Pete	Allen	M	AUTOMOTIVE	3
E083	Patrick	Voltz	M	HEALTHCARE	3
E001	Arthur	Black	M	ALL	5
E583	Janet	Hale	F	RETAIL	3
E612	Tracy	Norris	F	RETAIL	2

The Action Output pane shows the query execution details:

#	Time	Action	Message	Duration / Fetch
1	11:49:52	SELECT e.EMP_ID, e.FIRST_NAME, e.LAST_NAME, e.GENDER, e.DEPT AS DEPAR...	6 row(s) returned	0.000 sec / 0.000 sec

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 SQL Developer interface with a query window open. The query is as follows:

```

37
38
39 -- Query to list employees from the healthcare department
40 • SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT AS DEPARTMENT
41 FROM emp_record_table
42 WHERE DEPT = 'healthcare';
43 UNION
44 -- Query to list employees from the finance department
45 SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT AS DEPARTMENT
46 FROM emp_record_table
47 WHERE DEPT = 'finance';
48
49

```

The result grid shows the following data:

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

The Action Output pane shows the query execution details:

#	Time	Action	Message	Duration / Fetch
1	12:00:36	SELECT EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPT AS DEPARTMENT ...	7 row(s) returned	0.000 sec / 0.000 sec

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.

=> **SELECT**

```
e.EMP_ID,
e.FIRST_NAME,
e.LAST_NAME,
e.ROLE,
e.DEPT AS DEPARTMENT,
e.EMP_RATING,
max_ratings.max_emp_rating AS MAX_EMP_RATING_FOR_DEPT
```

**FROM**

```
emp_record_table e
```

**JOIN (**

```
SELECT DEPT, MAX(EMP_RATING) AS max_emp_rating
FROM emp_record_table
GROUP BY DEPT
```

**) max\_ratings ON e.DEPT = max\_ratings.DEPT**

**ORDER BY e.DEPT, e.EMP\_ID;**

The screenshot shows a SQL IDE interface. On the left is a 'SCHEMAS' panel with a tree view of database objects. The main area is a query editor with the following SQL query:

```

SELECT
  e.EMP_ID,
  e.FIRST_NAME,
  e.LAST_NAME,
  e.ROLE,
  e.DEPT AS DEPARTMENT,
  e.EMP_RATING,
  max_ratings.max_emp_rating AS MAX_EMP_RATING_FOR_DEPT
FROM
  emp_record_table e
JOIN (
  SELECT DEPT, MAX(EMP_RATING) AS max_emp_rating
  FROM emp_record_table
  GROUP BY DEPT
) max_ratings ON e.DEPT = max_ratings.DEPT
ORDER BY e.DEPT, e.EMP_ID;

```

Below the query editor is a 'Result Grid' showing 19 rows of data. The columns are EMP\_ID, FIRST\_NAME, LAST\_NAME, ROLE, DEPARTMENT, EMP\_RATING, and MAX\_EMP\_RATING\_FOR\_DEPT. The data is grouped by department: AUTOMOTIVE, FINANCE, HEALTHCARE, and RETAIL.

EMP_ID	FIRST_NAME	LAST_NAME	ROLE	DEPARTMENT	EMP_RATING	MAX_EMP_RATING_FOR_DEPT
E428	Pete	Allen	MANAGER	AUTOMOTIVE	4	5
E532	Claire	Brennan	ASSOCIATE DATA SCIENTIST	AUTOMOTIVE	1	5
E005	Eric	Hoffman	LEAD DATA SCIENTIST	FINANCE	3	4
E103	Emily	Grove	MANAGER	FINANCE	4	4
E403	Steve	Hoffman	ASSOCIATE DATA SCIENTIST	FINANCE	3	4
E052	Dianna	Wilson	SENIOR DATA SCIENTIST	HEALTHCARE	5	5
E057	Dorothy	Wilson	SENIOR DATA SCIENTIST	HEALTHCARE	1	5
E083	Patrick	Voltz	MANAGER	HEALTHCARE	5	5
E505	Chad	Wilson	ASSOCIATE DATA SCIENTIST	HEALTHCARE	2	5
E245	Nian	Zhen	SENIOR DATA SCIENTIST	RETAIL	2	4
E260	Roy	Collins	SENIOR DATA SCIENTIST	RETAIL	3	4
E478	David	Smith	ASSOCIATE DATA SCIENTIST	RETAIL	4	4
E583	Janet	Hale	MANAGER	RETAIL	2	4
FA17	Tracy	Norris	MANAGER	RETAIL	4	4

At the bottom, the 'Output' panel shows the execution log with two entries:

- 1 12:00:36 SELECT EMP\_ID, FIRST\_NAME, LAST\_NAME, GENDER, DEPT AS DEPARTMENT ... 7 row(s) returned 0.000 sec / 0.000 sec
- 2 12:05:40 SELECT e.EMP\_ID, e.FIRST\_NAME, e.LAST\_NAME, e.ROLE, e.DEPT A... 19 row(s) returned 0.047 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 SQL Developer interface with a query window containing the following SQL code:

```

64 ORDER BY e.DEPT, e.EMP_ID;
65
66
67 SELECT ROLE, MIN(SALARY) AS MIN_SALARY, MAX(SALARY) AS MAX_SALARY
68 FROM emp_record_table
69 GROUP BY ROLE;
70
71

```

The Results pane displays the following data:

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

The Action Output pane shows the execution of the query, indicating that 6 rows were returned.

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 SQL Developer interface with a query window containing the following SQL code:

```

70
71 SELECT
72 EMP_ID,
73 FIRST_NAME,
74 LAST_NAME,
75 GENDER,
76 DEPT AS DEPARTMENT,
77 EXP,
78 RANK() OVER (ORDER BY EXP DESC) AS EXPERIENCE_RANK
79 FROM
80 emp_record_table;
81

```

The Results pane displays the following data:

EMP_ID	FIRST_NAME	LAST_NAME	GENDER	DEPARTMENT	EXP	EXPERIENCE_RANK
E583	Janet	Hale	F	RETAIL	14	3
E612	Tracy	Norris	F	RETAIL	13	6
E010	William	Butler	M	AUTOMOTIVE	12	7
E005	Eric	Hoffman	M	FINANCE	11	8
E057	Dorothy	Wilson	F	HEALTHCARE	9	9
E204	Karene	Nowak	F	AUTOMOTIVE	8	10
E260	Roy	Collins	M	RETAIL	7	11
E052	Dianna	Wilson	F	HEALTHCARE	6	12
E045	Nian	Zhen	M	RETAIL	6	12

The Action Output pane shows the execution of the query, indicating that 19 rows were 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.

=>code:

**CREATE VIEW high\_salary\_employees\_view AS**

**SELECT**

**EMP\_ID,**

**FIRST\_NAME,**

**LAST\_NAME,**

**GENDER,**

**DEPT AS DEPARTMENT,**

**COUNTRY,**

**SALARY**

FROM  
emp\_record\_table  
WHERE  
SALARY > 6000;

The screenshot shows the SQL Developer interface. The left pane displays the 'SCHEMAS' tree with the 'employee' schema selected. The main query window contains the following SQL code:

```

CREATE VIEW high_salary_employees_view AS
SELECT
  EMP_ID,
  FIRST_NAME,
  LAST_NAME,
  GENDER,
  DEPT AS DEPARTMENT,
  COUNTRY,
  SALARY
FROM
  emp_record_table
WHERE
  SALARY > 6000;
SELECT * FROM high_salary_employees_view;

```

The 'Result Grid' shows the following data:

EMP_ID	FIRST_NAME	LAST_NAME	GENDER	DEPARTMENT	COUNTRY	SALARY
E103	Emily	Grove	F	FINANCE	CANADA	10500
E204	Karene	Nowak	F	AUTOMOTIVE	GERMANY	7500
E245	Nian	Zhen	M	RETAIL	CHINA	6500
E260	Roy	Collins	M	RETAIL	INDIA	7000
E428	Pete	Allen	M	AUTOMOTIVE	GERMANY	11000
E583	Janet	Hale	F	RETAIL	COLOMBIA	10000
E612	Tracy	Norris	F	RETAIL	INDIA	8500

The 'Output' pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
4	12:31:35	CREATE VIEW high_salary_employees_view AS SELECT	EMP_ID, FIRST_NAME...	0 row(s) affected 0.171 sec
5	12:31:55	SELECT * FROM high_salary_employees_view LIMIT 0, 1000	12 row(s) returned	0.016 sec / 0.000 sec

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 SQL Developer interface. The left pane displays the 'SCHEMAS' tree with the 'employee' schema selected. The main query window contains the following SQL code:

```

WHERE
  SALARY > 6000;
SELECT * FROM high_salary_employees_view;

SELECT EMP_ID, FIRST_NAME, LAST_NAME, EXP
FROM emp_record_table
WHERE EXP > 10;

```

The 'Result Grid' shows the following data:

EMP_ID	FIRST_NAME	LAST_NAME	EXP
E001	Arthur	Black	20
E005	Eric	Hoffman	11
E010	William	Butler	12
E083	Patrick	Voltz	15
E103	Emily	Grove	14
E428	Pete	Allen	14
E583	Janet	Hale	14

The 'Output' pane shows the execution log:

#	Time	Action	Message	Duration / Fetch
5	12:31:55	SELECT * FROM high_salary_employees_view LIMIT 0, 1000	12 row(s) returned	0.016 sec / 0.000 sec
6	12:37:19	SELECT EMP_ID, FIRST_NAME, LAST_NAME, EXP FROM emp_record_table WHERE	8 row(s) returned	0.000 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.

The screenshot displays the SQL Server Enterprise Manager interface. The left pane shows the 'SCHEMAS' tree with the 'employee' schema selected. The central pane shows the SQL script for creating a stored procedure named 'GetEmployeesWithExperience()'. The script includes a 'DELIMITER //' statement, a 'CREATE PROCEDURE' statement, a 'BEGIN' block containing a 'SELECT \* FROM emp\_record\_table WHERE EXP > 3;' query, an 'END //' statement, and a 'CALL GetEmployeesWithExperience();' statement. The right pane shows the 'Result Grid' with 15 rows of employee data. The bottom pane shows the 'Output' window with a message indicating that 15 rows were returned by the stored procedure call.

```
105 DELIMITER //  
106 CREATE PROCEDURE GetEmployeesWithExperience()  
107 BEGIN  
108     SELECT *  
109     FROM emp_record_table  
110     WHERE EXP > 3;  
111 END //  
112 DELIMITER ;  
113  
114 CALL GetEmployeesWithExperience();  
115
```

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
E052	Dianna	Wilson	F	SENIOR DATA SCIENTIST	HEALTHCARE	6	CANADA	NORTH AMERICA	5500	5	E083	P103
E057	Dorothy	Wilson	F	SENIOR DATA SCIENTIST	HEALTHCARE	9	USA	NORTH AMERICA	7700	1	E083	P302
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

Result 30 x

Output

#	Time	Action	Message	Duration / Fetch
1	12:42:48	CALL GetEmployeesWithExperience()	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 shows the SQL Developer interface with a script editor containing the following SQL code:

```
CREATE FUNCTION GetJobProfile(EXP INT) RETURNS VARCHAR(50) DETERMINISTIC
BEGIN
  DECLARE job_profile VARCHAR(50);
  CASE
    WHEN EXP <= 2 THEN
      SET job_profile = 'JUNIOR DATA SCIENTIST';
    WHEN EXP <= 5 THEN
      SET job_profile = 'ASSOCIATE DATA SCIENTIST';
    WHEN EXP <= 10 THEN
      SET job_profile = 'SENIOR DATA SCIENTIST';
    WHEN EXP <= 12 THEN
      SET job_profile = 'LEAD DATA SCIENTIST';
    ELSE
      SET job_profile = 'MANAGER';
  END CASE;
  RETURN job_profile;
END //
```

The Output window shows the execution results:

#	Time	Action	Message	Duration / Fetch
9	20:41:06	DROP FUNCTION 'employee'. 'GetJobProfile'	0 row(s) affected	0.657 sec
10	20:41:35	CREATE FUNCTION GetJobProfile(EXP INT) RETURNS VARCHAR(50) DETERMINISTIC	0 row(s) affected	0.172 sec

### Usage Example:

The screenshot shows the SQL Developer interface with a query editor containing the following SQL code:

```
SELECT EMP_ID, FIRST_NAME, LAST_NAME, EXP, GetJobProfile(EXP) AS JOB_PROFILE
FROM emp_record_table;
```

The Result Grid shows the following data:

EMP_ID	FIRST_NAME	LAST_NAME	EXP	JOB_PROFILE
E001	Arthur	Black	20	MANAGER
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
E083	Patrick	Voltz	15	MANAGER
E103	Emily	Grove	14	MANAGER
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
E428	Pete	Allen	14	MANAGER
E478	David	Smith	3	ASSOCIATE DATA SCIENTIST
E505	Chad	Wilson	5	ASSOCIATE DATA SCIENTIST
E532	Claire	Brennan	3	ASSOCIATE DATA SCIENTIST
E582	Tamara	Hale	14	MANAGER

The Output window shows the execution results:

#	Time	Action	Message	Duration / Fetch
10	20:41:35	CREATE FUNCTION GetJobProfile(EXP INT) RETURNS VARCHAR(50) DETERMINISTIC	0 row(s) affected	0.172 sec

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.

-- Step 1: Check the execution plan

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

- classicmodels
- dba\_june
- dsa1
- employee
  - Tables
    - data\_science\_team
    - emp\_record\_table
    - proj\_table
  - Views
  - Stored Procedures
- pdp\_dai
  - Tables
  - Views
  - Stored Procedures

Administration Schemas

Information

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

Object Info Session

day1\* sq day3 day4 day5\* sql employee project\*

Limit to 1000 rows

```

115
116
117 • USE employee;
118
119 • SHOW TABLES;
120
121 -- Step 1: Check the execution plan
122 • EXPLAIN SELECT *
123   FROM emp_record_table
124   WHERE FIRST_NAME = 'Eric';

```

Result Grid

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	emp_record_table		ALL					19	10.00	Using where

Result 32 x

Output

Action Output

#	Time	Action	Message	Duration / Fetch
15	12:59:15	SHOW TABLES	5 row(s) returned	0.015 sec / 0.000 sec
16	13:03:04	EXPLAIN SELECT * FROM emp_record_table WHERE FIRST_NAME = 'Eric'	1 row(s) returned	0.000 sec / 0.000 sec

## -- Step 2: Create the index

Navigator

SCHEMAS

Filter objects

- LAST\_NAME
- GENDER
- ROLE
- DEPT
- EXP
- COUNTRY
- CONTINENT
- SALARY
- EMP\_RATING
- MANAGER\_ID
- PROJ\_ID
- Indexes
  - idx\_first\_name
- Foreign Keys
- Triggers

Administration Schemas

Information

No object selected

day1\* sq day3 day4 day5\* sql employee project\*

Limit to 1000 rows

```

114 • CALL GetEmployeesWithExperience();
115
116
117 • USE employee;
118
119 • SHOW TABLES;
120
121 -- Step 1: Check the execution plan
122 • EXPLAIN SELECT *
123   FROM emp_record_table
124   WHERE FIRST_NAME = 'Eric';
125
126
127
128 -- Step 2: Create the index
129 • ALTER TABLE emp_record_table
130   MODIFY COLUMN FIRST_NAME VARCHAR(255);
131
132 • CREATE INDEX idx_first_name ON emp_record_table (FIRST_NAME(50));
133

```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
29	17:57:47	ALTER TABLE emp_record_table MODIFY COLUMN FIRST_NAME VARCHAR(255)	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.172 sec
30	17:58:03	CREATE INDEX idx_first_name ON emp_record_table (FIRST_NAME(50))	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0	0.797 sec

## -- Step 3: Verify index creation

Navigator

SCHEMAS

Filter objects

employee

- data\_science\_team
- emp\_record\_table
- proj\_table
- Views
- Stored Procedures
- Functions
- pdp\_da1
- Tables
- Views
- Stored Procedures
- Functions
- pdp\_joins

Administration Schemas

Information

Table: data\_science\_team

Columns:

EMP_ID	text
FIRST_NAME	text
LAST_NAME	text
GENDER	text
ROLE	text
DEPT	text
EXP	int
COUNTRY	text
CONTINENT	text

Object Info Session

day1\* sq day3 day4 day5\* sql employee project\*

Limit to 1000 rows

```

126
127
128 -- Step 2: Create the index
129 ALTER TABLE emp_record_table
130 MODIFY COLUMN FIRST_NAME VARCHAR(255);
131
132 CREATE INDEX idx_first_name ON emp_record_table (FIRST_NAME(50));
133
134 -- Step 3: Verify index creation
135 SHOW INDEX FROM emp_record_table;
136

```

Result Grid

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Exp
emp_record_table	1	idx_first_name	1	FIRST_NAME	A	19	50	NO	YES	BTREE			YES	NO

Result 35

Output

Action Output

#	Time	Action	Message	Duration / Fetch
31	18:00:18	SHOW INDEX FROM employee	Error Code: 1146. Table 'employee.employee' doesn't exist	0.000 sec
32	18:00:32	SHOW INDEX FROM emp_record_table	1 row(s) returned	0.000 sec / 0.000 sec

## -- Step 4: Re-check the execution plan

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

employee

- data\_science\_team
- emp\_record\_table
- Columns
- EMP\_ID
- FIRST\_NAME
- LAST\_NAME
- GENDER
- ROLE
- DEPT
- EXP
- COUNTRY
- CONTINENT
- SALARY

Administration Schemas

Information

Table: emp\_record\_table

Columns:

EMP_ID	text
FIRST_NAME	varchar(255)
LAST_NAME	text
GENDER	text
ROLE	text
DEPT	text
EXP	int
COUNTRY	text
CONTINENT	text
SALARY	int
EMP_RATING	int
MANAGED_IN	text

Object Info Session

day1\* sq day3 day4 day5\* sql employee project\*

Limit to 1000 rows

```

132 CREATE INDEX idx_first_name ON emp_record_table (FIRST_NAME(50));
133
134 -- Step 3: Verify index creation
135 SHOW INDEX FROM emp_record_table;
136
137 -- Step 4: Re-check the execution plan
138 EXPLAIN SELECT *
139 FROM emp_record_table
140 WHERE FIRST_NAME = 'Eric';
141

```

Result Grid

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	emp_record_table	NO	ref	idx_first_name	idx_first_name	203	const	1	100.00	Using where

Result 36

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	18:02:31	EXPLAIN SELECT * FROM emp_record_table WHERE FIRST_NAME = 'Eric'	1 row(s) returned	0.000 sec / 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).

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHMAS

Filter objects

employee

Tables

data\_science\_team

emp\_record\_table

Columns

EMP\_ID

FIRST\_NAME

LAST\_NAME

GENDER

ROLE

DEPT

EXP

COUNTRY

CONTINENT

SALARY

Table: emp\_record\_table

Columns:

EMP\_ID text

FIRST\_NAME varchar(255)

LAST\_NAME text

GENDER text

ROLE text

DEPT text

EXP int

COUNTRY text

CONTINENT text

SALARY int

EMP\_RATING int

MANAGED TO text

Object Info Session

day1\* sq day3 day4 day5\* sql employee project\*

Limit to 1000 rows

142

143

144

145

146

147

148

149

150

151

152

SELECT

EMP\_ID,

FIRST\_NAME,

LAST\_NAME,

SALARY,

EMP\_RATING,

0.05 \* SALARY \* EMP\_RATING AS BONUS

FROM

emp\_record\_table;

Result Grid

Filter Rows:

Export:

Wrap Cell Contents

EMP_ID	FIRST_NAME	LAST_NAME	SALARY	EMP_RATING	BONUS
E103	Emily	Grove	10500	4	2100.00
E204	Karene	Nowak	7500	5	1875.00
E245	Nian	Zhen	6500	2	650.00
E260	Roy	Collins	7000	3	1050.00
E403	Steve	Hoffman	5000	3	750.00
E428	Pete	Allen	11000	4	2200.00
E478	David	Smith	4000	4	800.00

Result 37

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	18:02:31	EXPLAIN SELECT * FROM emp_record_table WHERE FIRST_NAME = 'Eric'	1 row(s) returned	0.000 sec / 0.000 sec
2	18:06:07	SELECT EMP_ID, FIRST_NAME, LAST_NAME, SALARY, EMP_RATING...	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.

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHMAS

Filter objects

employee

Tables

data\_science\_team

emp\_record\_table

Columns

EMP\_ID

FIRST\_NAME

LAST\_NAME

GENDER

ROLE

DEPT

EXP

COUNTRY

CONTINENT

SALARY

Table: emp\_record\_table

Columns:

EMP\_ID text

FIRST\_NAME varchar(255)

LAST\_NAME text

GENDER text

ROLE text

DEPT text

EXP int

COUNTRY text

CONTINENT text

SALARY int

EMP\_RATING int

MANAGED TO text

Object Info Session

day1\* sq day3 day4 day5\* sql employee project\*

Limit to 1000 rows

152

153

154

155

156

157

158

159

160

161

162

163

SELECT

CONTINENT,

COUNTRY,

AVG(SALARY) AS AVG\_SALARY

FROM

emp\_record\_table

GROUP BY

CONTINENT,

COUNTRY;

Result Grid

Filter Rows:

Export:

Wrap Cell Contents

CONTINENT	COUNTRY	AVG_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

Result 38

Output

Action Output

#	Time	Action	Message	Duration / Fetch
2	18:06:07	SELECT EMP_ID, FIRST_NAME, LAST_NAME, SALARY, EMP_RATING...	19 row(s) returned	0.000 sec / 0.000 sec
3	18:07:50	SELECT CONTINENT, COUNTRY, AVG(SALARY) AS AVG_SALARY FROM ...	7 row(s) returned	0.016 sec / 0.000 sec