Peer Learning - RDBMS Assignment

Question-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.

Karan's Solution:

```
create database employee;
use employee;
```

Akshay's Solution:

```
create database employee;
use employee;
```

Conclusion: Both their solutions are the same as mine. The creation of tables are also the same.

Question-2: Create an ER diagram for the given employee database.
Conclusion: Both Karan's and akshay's solutions are the same as mine. The relationship between emp_record_table and proj_table is many to one.

Question-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.
Karan's Solution:

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

Akshay's Solution:

```
SELECT emp_id, first_name, last_name, gender, dept
FROM emp_record_table;
```

Conclusion: All three of our solutions are the same. The only difference is I used aliases for each column in the **select** statement.

Question-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

Conclusion: All of our solutions are again the same. Now also the only difference is I used aliases for each column in the **select** statement.

Question-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.

Common solution:

```
SELECT CONCAT(first_name, ' ', last_name) as NAME
FROM emp_record_table
WHERE dept='Finance';
```

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

Karan's Solution:

```
SELECT mgr.FIRST_NAME,COUNT(*) as number_of_reporters
FROM emp_record_table emp JOIN emp_record_table mgr
ON emp.MANAGER_ID=mgr.EMP_ID
GROUP BY mgr.FIRST_NAME;
```

Akshay's Solution:

```
SELECT mgr.emp_id, mgr.first_name, mgr.last_name, COUNT(e.emp_id)
FROM emp_record_table AS e INNER JOIN emp_record_table AS mgr
ON e.manager_id = mgr.emp_id
GROUP BY mgr.emp_id, mgr.first_name, mgr.last_name;
```

Conclusion: I used a derived table as a second table which will return manager Ids and number of employees reporting to him and join with the employee table to get desired output. But karan and akshay got the same result by self joining the employee record table.

Question-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.

Conclusion: All three of our solutions are the same as the question demanded to use only one approach to solve the query. I used aliases in the **select** statement.

Question-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.

Common solution:

```
SELECT ert.EMP_ID, ert.FIRST_NAME, ert.LAST_NAME, ert.ROLE, ert.DEPT ,
ert.EMP_RATING,
MAX(ert.EMP_RATING) OVER(PARTITION BY ert.DEPT) as MAX_RATING_BY_DEPT
FROM emp_record_table ert;
```

Conclusion: The only difference between both of their solutions from mine is that I used aliases in **select** statements which are very readable in the output table.

Question-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.
Karan's Solution:

```
SELECT DISTINCT ROLE,
MAX(ert.SALARY) OVER(PARTITION BY ert.ROLE) AS max_salary,
MIN(ert.SALARY) OVER(PARTITION BY ert.ROLE) AS min_salary
FROM emp_record_table ert;
```

Akshay's Solution:

```
SELECT MIN(salary) as min_sal_by_role, MAX(salary) as max_sal_by_role
FROM emp_record_table
GROUP BY role;
```

Conclusion: Karan's And I used window function to solve the query whereas Akshay used simple Group by statement to solve.

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

Karan's and Akshay's Solution:

```
SELECT EMP_ID,EXP,
DENSE_RANK() OVER(ORDER BY EXP DESC) AS rnk
FROM emp_record_table;
```

Conclusion: Both of them used the **Dense_rank** function to rank the employees, whereas I used the **Rank** function to solve.

Question-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.

Common Solution:

```
CREATE OR REPLACE VIEW emp_salary_greater_than_six AS
SELECT EMP_ID,FIRST_NAME,LAST_NAME,COUNTRY,SALARY FROM emp_record_table
WHERE SALARY > 6000;
select * from emp_salary_greater_than_six;
```

Conclusion: All of our solutions are the same, I extracted all the employee information whereas they only specified some columns to display in the view.

Question-12: Write a nested query to find employees with experience of more than ten years. Take data from the employee record table.
Conclusion: All of our solutions are the same.

Question-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.

Karan's and Akshay's Solution:

```
DELIMITER $$
CREATE PROCEDURE employeeDetails()
BEGIN
SELECT *
FROM emp_record_table
WHERE EXP > 3;
END $$
DELIMITER ;
CALL employeeDetails(); -- To call the procedure
```

Conclusion: I used the concept of cursors to display the details of employees whose experience is greater than 3. Where both of them used a direct select statement to get the result.

Question-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'.

Conclusion: All of our solutions are the same with minor implementation differences.

Question-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.

Common Solution:

```
CREATE
INDEX index_first_name
ON emp_record_table(FIRST_NAME);
SELECT *
FROM emp_record_table WHERE FIRST_NAME='Eric';
```

Conclusion: All of our three solutions are the same.

Question-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).

Common Solution:

```sql
SELECT EMP_ID,EMP_RATING,SALARY,((0.05 * SALARY)*EMP_RATING) AS BONUS
FROM emp_record_table;
```

Conclusion: All of our three solutions are the same as the question is quite straightforward.

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

Common Solution:

```sql
SELECT DISTINCT CONTINENT,
AVG(SALARY) OVER(PARTITION BY CONTINENT) AS avg_salary_by_continent,
COUNTRY,
AVG(SALARY) OVER(PARTITION BY COUNTRY) AS avg_salary_by_country FROM
emp_record_table;
```

Conclusion: All of our three solutions are the same where we all used window function to solve the query.