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.

Solution:

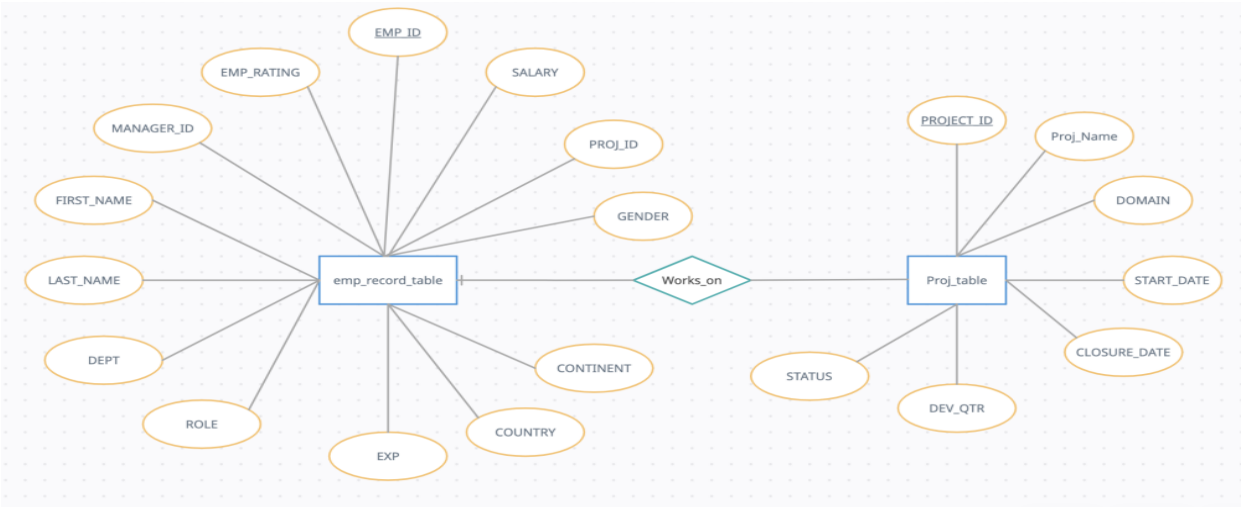
```
CREATE DATABASE employee;
USE employee;
```

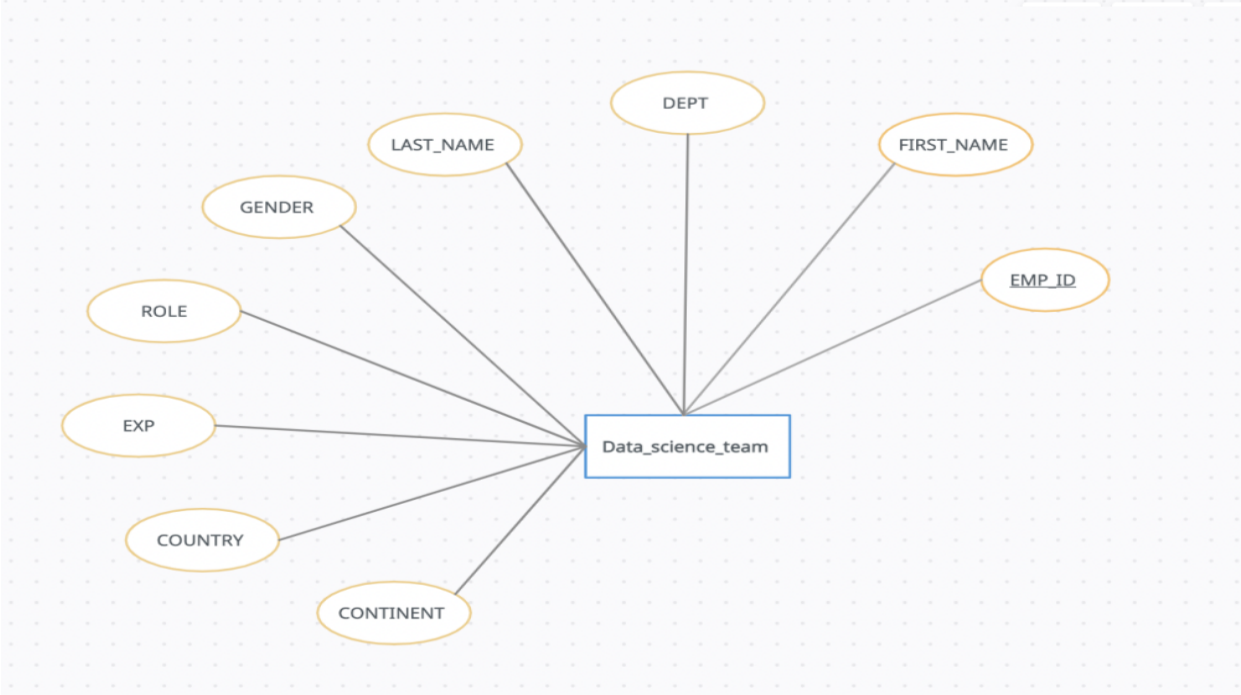
After creating the database I Imported the given csv files through GUI of the MYSQL Workbench.

Question 2:

Create an ER diagram for the given employee database:

Solution:-





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.

Solution:-

```
-- Fetching the required data columns from emp_record_table

SELECT ert.EMP_ID, ert.FIRST_NAME, ert.LAST_NAME, ert.GENDER,
ert.DEPT
FROM emp_record_table ert;
– Considering ert as an alias for emp_record_table
```

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

Solution:-

```
Query to fetch required data columns where employee rating(EMP_RATING) is
 LESS THAN 2

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

-- Considering ert as alias for emp_record_table

-- Query to fetch required data columns where employee rating(EMP_RATING) is
   GREATER THAN 4

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

-- Query to fetch required data columns where employee rating(EMP_RATING) is
   BETWEEN 2 AND 4
```

```
SELECT ert.EMP_ID, ert.FIRST_NAME, ert.LAST_NAME, ert.GENDER,
ert.DEPT , ert.EMP_RATING
FROM emp_record_table ert
WHERE ert.EMP_RATING between 2 and 4;
```

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.

Solution:-

```
SELECT CONCAT(ert.FIRST_NAME,' ',ert.LAST_NAME) as NAME
FROM emp_record_table ert
where ert.dept='FINANCE';
```

– Concatenating name and last name of employees with alias as name from emp_record_table where department is 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).

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;
```

-- Fetching the manager name(employees whom other employees are reporting to) by self joining the emp_record_table with itself as manager is also an employee with specific EMP_ID

– For first table i have given alias as emp to emp_record_table and considering the same table as second table i have given alias as mgr

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.

Solution:-

```
-- Union returns all the distinct rows between the data fetched by two select statements

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

– Here First select statements fetch all the records from the emp_record_table
  where department is Healthcare
– Union operator combines two select statements and gives distinct rows as
  result
– Here Second select statements fetch all the records from the
  emp_record_table where department is FINANCE
```

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.

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;
```

-- Fetching all the required columns asked from the emp_record_table
– Using window function to calculate MAX of emp_rating partition by department
FROM emp_record_table ert;(Considering ert as alias for emp_record_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.

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;
```

-- In this question we have been asked to fetch the minimum and maximum salary of the employees based on each role, so i have used window function to fetch the maximum and minimum using aggregate function max and min partition by ROLE(based on each role)

-- Used DISTINCT Keyword here because window funtion fetch the results in the form of frame,so duplicate data is returned based on the column fetched, so to avoid redundant data i have used DISTINCT Keyword

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

Solution:-

```
SELECT EMP_ID,EXP,
```

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

– Here I have used DENSE_RANK() function to assign ranks to the employees
  based on their experience
-- window function will assign ranks using DENSE_RANK order by experience
  from emp_record_table in descending order so that for highest exp it will rank 1
  and so on
```

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.

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;

-- Here I created a simple view called emp_salary_greater_than_six which
  fetches all the records where salary is greater than 6000
-- I have used create or replace because if view already exists than replace with
  the current content but if it does not exists than create a view with this content
```

Question 12:

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

Solution:-

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

-- In this particular question to use concept of nested query I have first fetched the exp from the emp_record_table with the condition that exp > 10 which will return all the exp greater than 10 in the WHERE CLAUSE of the outer select statement and from outer select statement I am fetching all the records where exp matches with the result of the inner query and hence fetch employees with experience more than 10 years

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.

Solution:-

-- Procedure is one of the subprogram where we can store our query to reuse later as and when required

```
DELIMITER $$
CREATE PROCEDURE employeeDetails()
BEGIN
   SELECT * FROM
   emp_record_table
   WHERE EXP > 3;
END $$
DELIMITER ;
```

CALL employeeDetails(); – To call the procedure

-- Here I have created Procedure name employeeDetails() which when called always returns the details of employees with experience greater than 3

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'.

Solution:-

```
DELIMITER $$
CREATE FUNCTION checkOrganizationStandard()
RETURNS tinyint(1)      -- Return type tinyint(1) as function return boolean
                        – value(True(1)– or False(0))

DETERMINISTIC           -- Function is Deterministic because it always gives the
                           constant output for fixed set of inputs

BEGIN
-- Declare finished variable which will be used later while handling exception to
– set it to 1 to exit the loop
DECLARE finished INTEGER DEFAULT 0;

-- Declaring v_exp variable to fetch exp column data from data_science_team
  while creating cursor
DECLARE v_exp INTEGER;


-- Declaring v_role variable to fetch role column data from data_science_team
  while creating cursor
DECLARE v_role VARCHAR(50);


-- Creating the cursor with name expcursor which fetches exp and role from
```

```
  data_science_team
DECLARE expcursor
CURSOR FOR SELECT exp,role
                    FROM data_science_team;


-- Declaring the handler to avoid the exception when cursor reaches the end
  of the row and their is no row to fetch

 -- Setting finished to 1 which keeps a track of whether exception is generated
   or not, when it is set to 1we exit the loop and no more fetching is done

DECLARE CONTINUE HANDLER FOR NOT FOUND
       SET finished = 1;

OPEN expcursor;                    -- To open the cursor

-- Loop that will fetch each row with the help of cursor and then check the desired
condition specified and accordingly gives the result

checkStandard:LOOP
-- Fetching the data from the cursor into v_exp and v_role variable
    FETCH expcursor INTO v_exp,v_role;

 -- whenever finished=1 condition satisfies we break the loop as their is no
    more rows to traverse
    IF finished = 1 THEN
       LEAVE checkStandard;
    END IF;

-- Used Case to check for organization's set standard as per the given question
-- I have used the CASE over normal If because:-
-- A simple CASE statement is more readable and efficient than an IF statement
  when you compare a single expression against a range of unique values.

    CASE v_exp
      WHEN v_exp <= 2 THEN
```

```
            IF v_role != 'JUNIOR DATA SCIENTIST' THEN
              RETURN FALSE;
            END IF;
          WHEN v_exp > 2 and v_exp<=5 THEN
            IF v_role != 'ASSOCIATE DATA SCIENTIST' THEN
              RETURN FALSE;
            END IF;
          WHEN v_exp > 5 and v_exp<=10 THEN
            IF v_role != 'SENIOR DATA SCIENTIST' THEN
              RETURN FALSE;                    '
            END IF;
          WHEN v_exp > 10 and v_exp<=12 THEN
            IF v_role != 'LEAD DATA SCIENTIST' THEN
              RETURN FALSE;
            END IF;
          ELSE
            IF v_exp > 12 and v_exp<=16 AND v_role != 'MANAGER' THEN
              RETURN FALSE
            END IF;
        END CASE;

    END LOOP checkStandard;    -- Ending the loop

    CLOSE expcursor;           -- Closing the cursor

    -- Return True because if all the above condition in loop are not met none
       of the case returns false that means data is in accordance with
       organization's set standard
    RETURN TRUE;
    END $$
    DELIMITER ;


    – Created the procedure which will call the function created above and print
      the message 'Data matches the organization's set standard' if function
      returns true otherwise 'Data does not matches the organization's set
      standard' if it return false;
```

```
DELIMITER $$
CREATE PROCEDURE checkStandard(OUT res VARCHAR(20))
BEGIN

– Calling the function checkOrganizationStandard()
IF(checkOrganizationStandard()) THEN
   set res= 'Data matches the organization’s set standard';
ELSE
   SET res='Data does not matches the organization’s set standard';
END IF;
END $$


CALL checkStandard(@res);    – Calling the Procedure
SELECT @res as Result;
```

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.

Solution:-

```
CREATE
INDEX index_first_name
ON emp_record_table(FIRST_NAME);

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

-- Here I am creating the index on FIRSTNAME attribute of emp_record_table and then after i am fetching the record where first_name is 'Eric'

-- Performance Analysis
-- BEFORE CREATING INDEX THE QUERY TOOK 0.00061 sec
-- AFTER CREATING THE INDEX THE QUERY TOOK 0.00052 sec

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).

Solution:-

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

-- Here getting the desired result by performing just the airthmetic operation
in select Statement

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

Solution:-

```
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;
```

-- Here I have used the window function to get the average salary by continent
as well as average salary by country