/* Below are the queries as per the questions in the dataset */

```
-- CREATING TABLES FROM THE PROVIDED DATASETS

CREATE TABLE emp_record_table(
        EMP_ID INT PRIMARY KEY,
   FIRST_NAME VARCHAR(50),
   LAST_NAME VARCHAR (50),
   GENDER VARCHAR(5),
   ROLE VARCHAR(50),
   DEPT VARCHAR(50),
   EXP INT,
   COUNTRY VARCHAR(50),
   CONTINENT VARCHAR(50),
   SALARY DECIMAL(10,2),
   EMP_RATING DECIMAL(5,2),
   MANAGER_ID INT,
   PROJ_ID INT
);

CREATE TABLE proj_table(
        PROJECT_ID INT PRIMARY KEY,
   PROJ_NAME VARCHAR(50),
   DOMAIN VARCHAR(50),
   START_DATE DATE,
   CLOSURE_DATE DATE,
   DEV_QTR VARCHAR(50),
   STATUS VARCHAR(50)
);

CREATE TABLE data_science_team(
        EMP_ID INT PRIMARY KEY,
   FIRST_NAME VARCHAR(50),
   LAST_NAME VARCHAR(50),
   GENDER VARCHAR(50),
   ROLE VARCHAR(50),
   DEPT VARCHAR(50),
   EXP INT,
   COUNTRY VARCHAR(50),
   CONTINENT VARCHAR(50)
);


-- LOADING DATA INTO THE TABLES

-- For emp_record_table
        LOAD DATA INFILE 'emp_record_table.csv'
```

```
        INTO TABLE emp_record_table
        FIELDS TERMINATED BY ','
        LINES TERMINATED BY '\n'
        IGNORE 1 LINES;

-- For proj_table
        LOAD DATA INFILE 'proj_table.csv'
        INTO TABLE proj_table
        FIELDS TERMINATED BY ','
        LINES TERMINATED BY '\n'
        IGNORE 1 LINES;

-- For data_science_team
        LOAD DATA INFILE 'data_science_team.csv'
        INTO TABLE data_science_team
        FIELDS TERMINATED BY ','
        LINES TERMINATED BY '\n'
        IGNORE 1 LINES;


        use project_sql;

-- WAQ Fetch EMP_ID, FIRST_NAME, LAST_NAME, GENDER, and DEPARTMENT from the
employee record table:
        SELECT emp_id, first_name, last_name, gender, dept
        FROM emp_record_table;

select * from emp_record_table;

-- WAQ Fetch EMP_ID, FIRST_NAME, LAST_NAME, GENDER, DEPARTMENT, and
EMP_RATING based on EMP_RATING conditions:
        select EMP_ID, FIRST_NAME, LAST_NAME, GENDER, dept, emp_rating from
        emp_record_table where emp_rating >4;
        select EMP_ID, FIRST_NAME, LAST_NAME, GENDER, dept, emp_rating from
        emp_record_table where emp_rating <2;
        select EMP_ID, FIRST_NAME, LAST_NAME, GENDER, dept, emp_rating from
        emp_record_table where emp_rating between 2 and 4;

-- WAQ Concatenate FIRST_NAME and LAST_NAME of employees in the Finance department:
        SELECT CONCAT(first_name, ' ', last_name) AS NAME
        FROM emp_record_table
        WHERE dept = 'Finance';

-- WAQ List employees who have someone reporting to them and show the number of
reporters:
        SELECT e1.emp_id, e1.first_name, e1.last_name, COUNT(e2.emp_id) AS
        num_reporters
```

```
    FROM emp_record_table e1
    JOIN emp_record_table e2 ON e1.emp_id = e2.manager_id
    GROUP BY e1.emp_id, e1.first_name, e1.last_name;

-- WAQ List employees from the healthcare and finance departments using UNION:
    SELECT emp_id, first_name, last_name, dept
    FROM emp_record_table
    WHERE dept = 'Healthcare'

    UNION

    SELECT emp_id, first_name, last_name, dept
    FROM emp_record_table
    WHERE dept = 'Finance';

-- WAQ Group employee details by dept with their ratings and max rating for the department:
    SELECT dept, emp_id, first_name, last_name, role, emp_rating, MAX(emp_rating)
    OVER (PARTITION BY dept) AS max_emp_rating
    FROM emp_record_table;

-- WAQ Calculate the minimum and maximum salary of employees in each role:
    SELECT role, MIN(salary) AS min_salary, MAX(salary) AS max_salary
    FROM emp_record_table
    GROUP BY role;

-- WAQ Assign ranks to each employee based on their experience:
    SELECT first_name, last_name, exp,
        RANK() OVER (ORDER BY exp DESC)
    FROM emp_record_table;

-- WAQ to Create a view for employees in various countries with a salary greater than six
thousand:
    create view high_salary_employees as select * from emp_record_table where salary >
    6000;

-- WAQ to Find employees with experience of more than ten years using a nested query:
    select emp_id, first_name, last_name, exp from emp_record_table where exp > 10;

-- WAQ to Create a stored procedure to retrieve details of employees with experience more than
three years:

    DELIMITER //

    CREATE PROCEDURE GetEmployeesByExperience()
    BEGIN
      SELECT emp_id, first_name, last_name, exp
      FROM emp_record_table
```

```
    WHERE exp > 3;
END //

DELIMITER ;
```

-- WAQ to Store function to check job profile based on experience standards:

```
DELIMITER //

CREATE FUNCTION GetJobProfile(exp INT) RETURNS VARCHAR(50)
DETERMINISTIC
BEGIN
    DECLARE profile VARCHAR(50);
    IF exp <= 2 THEN
        SET profile = 'JUNIOR DATA SCIENTIST';
    ELSEIF exp BETWEEN 2 AND 5 THEN
        SET profile = 'ASSOCIATE DATA SCIENTIST';
    ELSEIF exp BETWEEN 5 AND 10 THEN
        SET profile = 'SENIOR DATA SCIENTIST';
    ELSEIF exp BETWEEN 10 AND 12 THEN
        SET profile = 'LEAD DATA SCIENTIST';
    ELSEIF exp BETWEEN 12 AND 16 THEN
        SET profile = 'MANAGER';
    ELSE
        SET profile = 'OTHER';
    END IF;
    RETURN profile;
END //

DELIMITER ;
```

-- using the above function in a query

```
SELECT
    emp_id,
    first_name,
    last_name,
    role,
    GetJobProfile(exp) AS expected_profile
FROM
    emp_record_table
WHERE
    dept = 'Healthcare';
```

-- WAQ to Create an index to improve query performance for finding an employee named ‘Eric’:

```
CREATE INDEX idx_first_name ON emp_record_table(first_name(20));
```

-- Execution plan check

```
        EXPLAIN SELECT * FROM emp_record_table WHERE first_name = 'Eric';

-- WAQ to calculate the bonus for all employees based on their ratings and salaries:
        SELECT emp_id, first_name, last_name, salary, emp_rating,
        (salary * 0.05 * emp_rating) AS new_bonus
        FROM emp_record_table;

-- WAQ to Calculate the average salary distribution based on the continent and country:

        SELECT
           continent,
           country,
           AVG(salary) AS avg_salary
        FROM
           emp_record_table
        GROUP BY
           continent, country;


        select * from emp_record_table;


/* below are my self exploration queries for the same data set */

/*
-- CREATING TABLES FROM THE PROVIDED DATASETS

CREATE TABLE emp_record_table(
        EMP_ID INT PRIMARY KEY,
   FIRST_NAME VARCHAR(50),
   LAST_NAME VARCHAR (50),
   GENDER VARCHAR(5),
   ROLE VARCHAR(50),
   DEPT VARCHAR(50),
   EXP INT,
   COUNTRY VARCHAR(50),
   CONTINENT VARCHAR(50),
   SALARY DECIMAL(10,2),
   EMP_RATING DECIMAL(5,2),
   MANAGER_ID INT,
   PROJ_ID INT
);

CREATE TABLE proj_table(
        PROJECT_ID INT PRIMARY KEY,
   PROJ_NAME VARCHAR(50),
   DOMAIN VARCHAR(50),
```

```
    START_DATE DATE,
    CLOSURE_DATE DATE,
    DEV_QTR VARCHAR(50),
    STATUS VARCHAR(50)
);

CREATE TABLE data_science_team(
        EMP_ID INT PRIMARY KEY,
    FIRST_NAME VARCHAR(50),
    LAST_NAME VARCHAR(50),
    GENDER VARCHAR(50),
    ROLE VARCHAR(50),
    DEPT VARCHAR(50),
    EXP INT,
    COUNTRY VARCHAR(50),
    CONTINENT VARCHAR(50)
);


-- LOADING DATA INTO THE TABLES

-- For emp_record_table
LOAD DATA INFILE 'emp_record_table.csv'
INTO TABLE emp_record_table
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 LINES;

-- For proj_table
LOAD DATA INFILE 'proj_table.csv'
INTO TABLE proj_table
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 LINES;

-- For data_science_team
LOAD DATA INFILE 'data_science_team.csv'
INTO TABLE data_science_team
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 LINES;


-- performing analysis on the datasets.
-- employee record table, we will try to find the maximum salary from the salary fields in the
employee record table.
```

```
use project_sql;

select max(salary) as max_salary from emp_record_table;
-- exploring the data further by creating temp groups of employees based on their departments
select dept, COUNT(first_name) from emp_record_table group by dept;

-- to find the maximum salary from each department.
SELECT dept, MAX(salary) AS max_salary
FROM emp_record_table
GROUP BY dept;

select * from emp_record_table;

-- segregating the employees based on their experience and dept and salary, to see who is
being fairly compensated
SELECT EXP, dept, MAX(salary) AS max_salary
FROM emp_record_table
GROUP BY EXP, dept;

-- segregating the employees based on their gender to see their salary
select gender, exp, max(SALARY) as max_salary from emp_record_table group by gender, exp;

-- now we want to see the maximum salary that a female employee is being paid based on their
experience
select first_name, exp, max(salary) as max_salary from emp_record_table where gender = 'F'
group by first_name, EXP;

-- lets also take a look at the highest paid male employees
select first_name, exp, max(salary) as max_salary from emp_record_table where gender = 'M'
group by first_name, EXP;

-- let's create a base salary for comparison, by assuming the base saalry to be 10% of their
current salary
-- Add a new column base_salary
ALTER TABLE emp_record_table
ADD COLUMN base_salary DECIMAL(10, 2);

select * from emp_record_table;

-- Update base_salary as 10% of current salary
-- Disable safe update mode if not already disabled
SET SQL_SAFE_UPDATES = 0;

-- Update base_salary as 10% of current salary
UPDATE emp_record_table
SET base_salary = salary * 0.3;
```

```
select * from emp_record_table;

-- now that our base salary is created we need to make a comparison on the experience and
base salary to check for unfair compensation
SELECT  dept, EXP, max(base_salary) as max_base_salary
FROM emp_record_table group by dept, exp;


-- checking if the column was added to the existing table
select * from emp_record_table;

-- Step 1: Set the minimum base salary into a variable
SET @min_base_salary := (SELECT MIN(base_salary) FROM emp_record_table);

-- Step 2: Update base salaries using the variable
UPDATE emp_record_table
SET base_salary = base_salary / @min_base_salary;

-- since we encountered a truncate error for the base salary we need modify the column
ALTER TABLE emp_record_table
MODIFY COLUMN base_salary DECIMAL(12, 6);  -- Example: Increase to DECIMAL(12, 6)


-- verification of steps

SELECT emp_id, salary, base_salary
FROM emp_record_table;
-- we now have the contribution factor in name of base salary so let's change the name of the
column
ALTER TABLE emp_record_table
CHANGE COLUMN base_salary contribution_factor DECIMAL(10, 2);

-- now let's see who has contribution factor more than 3 and experience of more than 5 years
select * from emp_record_table where exp > 5 and contribution_factor >3;
-- let's filter this list further by seeing who has the employee rating of more than 3
SELECT * FROM emp_record_table WHERE contribution_factor > 3 AND exp > 5 and
emp_rating > 3;

-- people from this group who deserve a bonus and promotion are shown from the query below:
SELECT * FROM emp_record_table WHERE contribution_factor > 3 AND exp > 5 and
emp_rating > 3 and salary < 10000;

-- creating a promotion table where i can store the employee data who are qualified for
promotion
-- Step 1: Create the promotion table
CREATE TABLE promotion (
    emp_id INT PRIMARY KEY,
```

```
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    dept VARCHAR(50),
    salary DECIMAL(10, 2),
    exp INT,
    contribution_factor DECIMAL(10, 2)
);

-- adding employee rating column in the above table
alter table promotion add column emp_rating decimal(5,2);

-- checking the promotions table
select * from promotion;
-- Step 3: Insert filtered values into the promotion table
INSERT INTO promotion (emp_id, first_name, last_name, dept, salary, exp, contribution_factor,
emp_rating)
SELECT emp_id, first_name, last_name, dept, salary, exp, contribution_factor, emp_rating
FROM emp_record_table
WHERE contribution_factor > 3 AND exp > 5 AND emp_rating > 3 AND salary < 10000;
//*
describe emp_record_table;
describe promotion;

ALTER TABLE emp_record_table
MODIFY COLUMN emp_id VARCHAR(10);

ALTER TABLE promotion
MODIFY COLUMN emp_id VARCHAR(10);
-- the above steps were to correct the errors in the output
//*

-- ensuring all the jobs meets organization standards for profiles we will perform the following
function
select * from emp_record_table;
-- from output we can see that there are three roles we need to exclude to get the above result.

select * from emp_record_table where role not in ('Senior Data scientist','Manager','President');
-- calculate the bonsues from the salary based on emp_rating and contributing factor
select * from emp_record_table where emp_rating >3 and contribution_factor>2;

alter table emp_record_table add column bonus decimal(10,2);
-- checking the creation of the bonus column
select * from emp_record_table;
-- calculating and adding bonus data into the bonus column of the employee table
UPDATE emp_record_table
SET bonus = (salary * emp_rating * contribution_factor * 0.1)/10
WHERE emp_rating >= 3 AND contribution_factor > 2;
```

```
-- storing the best of these output values into the promotion table for the HR reference

SELECT *
FROM emp_record_table
WHERE emp_rating > 3
AND contribution_factor > 2;

select * from promotion;

-- finding out the employees and their respective projects

select e.emp_id, e.first_name, e.last_name,p.proj_name
from emp_record_table e
join proj_table p on e.PROJ_ID = p.PROJECT_ID;


select * from proj_table;
select * from emp_record_table;
-- join the project names in the emp record table
SELECT
   e.emp_id,
   e.first_name,
   e.last_name,
   (
      SELECT GROUP_CONCAT(p.proj_name ORDER BY p.proj_name SEPARATOR ', ')
      FROM proj_table p
      WHERE p.PROJECT_ID = e.proj_id
   ) AS projects
FROM
   emp_record_table e;

ALTER TABLE emp_record_table
ADD COLUMN project_names VARCHAR(1000);

UPDATE emp_record_table e
SET project_names = (
   SELECT GROUP_CONCAT(p.proj_name ORDER BY p.proj_name SEPARATOR ', ')
   FROM proj_table p
   WHERE p.PROJECT_ID = e.proj_id
);

-- now that we have the proejt names in the emp record table we can group by the proejct
names and see how many people are working on a particular project
SELECT project_names, COUNT(*) AS num_employees
FROM emp_record_table
GROUP BY project_names;
```

```
-- we need to alot new employees in the underemployed project
SELECT project_names, COUNT(*) AS num_employees
FROM emp_record_table
GROUP BY project_names
HAVING num_employees < 2;

UPDATE emp_record_table
SET emp_id = 5
WHERE project_names = 'Project B'
LIMIT 1;


-- add employee detail columns in the proj table
SELECT p.project_id, p.proj_name, e.emp_id, e.first_name, e.last_name, e.project_names
FROM proj_table p
LEFT JOIN emp_record_table e ON p.project_id = e.proj_id;


select * from proj_table;

-- creating a combined view
CREATE VIEW project_employee_details AS
SELECT p.PROJECT_ID, p.proj_name, e.emp_id, e.first_name, e.last_name, e.project_names
FROM proj_table p
LEFT JOIN emp_record_table e ON p.project_id = e.proj_id;

-- usage of the new created view
select * from Project_employee_details;

-- see the employee details w.r.t the projects grouped
SELECT proj_name, GROUP_CONCAT(first_name SEPARATOR ', ') AS employee_names
FROM project_employee_details
GROUP BY proj_name;

-- since the supply chain management project has 3 people assigned to it , we will try to assign
one of the employees from this project to the lung cancer project
-- selecting one employee from the suuppy chain management project
SELECT emp_id, first_name, last_name
FROM project_employee_details
WHERE proj_name LIKE '%Supply Chain Management%'
LIMIT 1;

-- assigning the employee to lung cancer project
UPDATE emp_record_table
SET project_names = REPLACE(project_names, 'Supply Chain Management', 'Early Detection
of Lung Cancer')
```

```
WHERE emp_id = 'E532';

select * from emp_record_table;

-- check if the new project  is assigned
SELECT project_names, COUNT(*) AS num_employees
FROM emp_record_table
GROUP BY project_names;

-- now we have managed to assign equal number of resources into all the active projects.

-- lets take a look at the data science team dataset
select * from data_science_team;

-- taking a look at datscience teams performance
select * from emp_record_table;
select d.emp_id, d.first_name,d.last_name,e.emp_rating
from data_science_team d
join emp_record_table e on d.EMP_ID = e.emp_id
order by e.emp_rating desc;

-- finding out the projects undertaken by datascience team

select d.emp_id, d.first_name,d.last_name,p.proj_name
from data_science_team d
join emp_record_table e on d.EMP_ID = e.EMP_ID
join proj_table p on e.PROJ_ID = p.PROJECT_ID;

-- we shall now group the data science team members according to their roles

SELECT
   role,
   GROUP_CONCAT(first_name SEPARATOR ', ') AS employees,
   dept
FROM data_science_team
GROUP BY role, dept;

-- to compare the fair compensation of the employees against their experience, we can perform
the following query

SELECT
   e.emp_id,
   e.first_name,
   e.last_name,
   e.dept,
   e.role,
   e.salary,
```

```
    e.exp,
    AVG(e.salary) OVER (PARTITION BY e.exp) AS avg_salary_by_exp,
    e.salary - AVG(e.salary) OVER (PARTITION BY e.exp) AS deviation_from_avg
FROM
    emp_record_table e
ORDER BY
    e.exp, e.salary DESC;

-- let us now see only the salary that is showing slight deviation from the averAGE Salary
-- creating columns to add the above calculation for further analysis
ALTER TABLE emp_record_table
ADD COLUMN avg_salary_by_exp DECIMAL(10,2),
ADD COLUMN deviation_from_avg DECIMAL(10,2);

select * from emp_record_table;

-- adding the above calculations into the emp record table as new columns
UPDATE emp_record_table e
JOIN (
    SELECT
        emp_id,
        AVG(salary) OVER (PARTITION BY exp) AS avg_salary_by_exp,
        salary - AVG(salary) OVER (PARTITION BY exp) AS deviation_from_avg
    FROM emp_record_table
) sub
ON e.emp_id = sub.emp_id
SET e.avg_salary_by_exp = sub.avg_salary_by_exp,
    e.deviation_from_avg = sub.deviation_from_avg;

-- viewing the people who have more deviation from the average salary

select * from emp_record_table where deviation_from_avg >0;

-- making the same comparison but now with employee rating above 3
select * from emp_record_table where deviation_from_avg >0 and emp_rating >3;
*/
```