SQL Queries_51

Database_Name: ORG

```
CREATE DATABASE ORG; //create database
SHOW DATABASES;
USE ORG;
```

Database_Table: Worker

```
CREATE TABLE Worker(
        WORKER_ID INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
   FIRST_NAME CHAR(25),
   LAST_NAME CHAR(25),
   SALARY INT(15),
   JOINING_DATE DATETIME,
   DEPARTMENT CHAR(25)
);

INSERT INTO Worker
        (WORKER_ID, FIRST_NAME, LAST_NAME, SALARY, JOINING_DATE, DEPARTMENT)
VALUES
                (001, 'Monika', 'Arora', 100000, '14-02-20 09.00.00', 'HR'),
                (002, 'Niharika', 'Verma', 80000, '14-06-11 09.00.00', 'Admin'),
                (003, 'Vishal', 'Singhal', 300000, '14-02-20 09.00.00', 'HR'),
                (004, 'Amitabh', 'Singh', 500000, '14-02-20 09.00.00', 'Admin'),
                (005, 'Vivek', 'Bhati', 500000, '14-06-11 09.00.00', 'Admin'),
                (006, 'Vipul', 'Diwan', 200000, '14-06-11 09.00.00', 'Account'),
                (007, 'Satish', 'Kumar', 75000, '14-01-20 09.00.00', 'Account'),
                (008, 'Geetika', 'Chauhan', 90000, '14-04-11 09.00.00', 'Admin');

SELECT * FROM WORKER;
```

Database_Table: Bonus

```
CREATE TABLE BONUS(
                WORKER_REF_ID INT,
    BONUS_AMOUNT INT(10),
    BONUS_DATE DATETIME,
    FOREIGN KEY (WORKER_REF_ID)
                        REFERENCES WORKER(WORKER_ID)
        ON DELETE CASCADE

);
```

```
INSERT INTO Bonus
        (WORKER_REF_ID, BONUS_AMOUNT, BONUS_DATE) VALUES
                (001, 5000, '16-02-20'),
                (002, 3000, '16-06-11'),
                (003, 4000, '16-02-20'),
                (001, 4500, '16-02-20'),
                (002, 3500, '16-06-11');

SELECT * FROM BONUS;
```

Database_Table: Title

```
CREATE TABLE Title (
        WORKER_REF_ID INT,
        WORKER_TITLE CHAR(25),
        AFFECTED_FROM DATETIME,
        FOREIGN KEY (WORKER_REF_ID)
                REFERENCES Worker(WORKER_ID)
    ON DELETE CASCADE
);
INSERT INTO Title
        (WORKER_REF_ID, WORKER_TITLE, AFFECTED_FROM) VALUES
 (001, 'Manager', '2016-02-20 00:00:00'),
 (002, 'Executive', '2016-06-11 00:00:00'),
 (008, 'Executive', '2016-06-11 00:00:00'),
 (005, 'Manager', '2016-06-11 00:00:00'),
 (004, 'Asst. Manager', '2016-06-11 00:00:00'),
 (007, 'Executive', '2016-06-11 00:00:00'),
 (006, 'Lead', '2016-06-11 00:00:00'),
 (003, 'Lead', '2016-06-11 00:00:00');

SELECT * FROM TITLE;
```

Question and Answer:

-- Q-1. Write an SQL query to fetch “FIRST_NAME” from Worker table using the alias name as <WORKER_NAME>.

```
SELECT FIRST_NAME AS WORKER_NAME FROM WORKER;
```

-- Q-2. Write an SQL query to fetch “FIRST_NAME” from Worker table in upper case.

```
SELECT UPPER(FIRST_NAME) FROM WORKER;
```

-- Q-3. Write an SQL query to fetch unique values of DEPARTMENT from Worker table.

```
SELECT DISTINCT DEPARTMENT FROM WORKER;
SELECT DEPARTMENT FROM WORKER GROUP BY DEPARTMENT;
```

-- Q-4. Write an SQL query to print the first three characters of FIRST_NAME from Worker table.

```
SELECT substring(FIRST_NAME, 1, 3) FROM WORKER;
```

-- Q-5. Write an SQL query to find the position of the alphabet ('b') in the first name column 'Amitabh' from Worker table.

```
SELECT INSTR(FIRST_NAME, 'B') FROM WORKER WHERE FIRST_NAME='Amitabh';
```

-- Q-6. Write an SQL query to print the FIRST_NAME from Worker table after removing white spaces from the right side.

```
SELECT RTRIM(FIRST_NAME) FROM WORKER;
```

-- Q-7. Write an SQL query to print the DEPARTMENT from Worker table after removing white spaces from the left side.

```
SELECT LTRIM(DEPARTMENT) FROM WORKER;
```

-- Q-8. Write an SQL query that fetches the unique values of DEPARTMENT from Worker table and prints its length.

```
SELECT DISTINCT DEPARTMENT, LENGTH(DEPARTMENT) FROM WORKER;
```

-- Q-9. Write an SQL query to print the FIRST_NAME from Worker table after replacing 'a' with 'A'.

```
SELECT REPLACE(FIRST_NAME, 'a', 'A') FROM WORKER;
```

-- Q-10. Write an SQL query to print the FIRST_NAME and LAST_NAME from Worker table into a single column COMPLETE_NAME.
-- A space char should separate them.

```
SELECT CONCAT(FIRST_NAME, ' ', LAST_NAME) AS COMPLETE_NAME FROM WORKER;
```

-- Q-11. Write an SQL query to print all Worker details from the Worker table order by FIRST_NAME Ascending.

```
SELECT * FROM WORKER ORDER BY FIRST_NAME ASC;
SELECT FIRST_NAME FROM WORKER ORDER BY FIRST_NAME ASC;
```

-- Q-12. Write an SQL query to print all Worker details from the Worker table order by
-- FIRST_NAME Ascending and DEPARTMENT Descending.

```
SELECT * FROM WORKER ORDER BY FIRST_NAME , DEPARTMENT DESC;
```

-- Q-13. Write an SQL query to print details for Workers with the first name as "Vipul" and "Satish" from Worker table.

```
SELECT * FROM WORKER WHERE FIRST_NAME IN( 'Vipul', 'Satish');
```

-- Q-14. Write an SQL query to print details of workers excluding first names, "Vipul" and "Satish" from Worker table.

SELECT * FROM WORKER WHERE FIRST_NAME NOT IN('Vipul', 'Satish');

-- Q-15. Write an SQL query to print details of Workers with DEPARTMENT name as "Admin*".
SELECT * FROM WORKER WHERE DEPARTMENT = 'Admin';
SELECT * FROM WORKER WHERE DEPARTMENT LIKE 'Admin%';

-- Q-16. Write an SQL query to print details of the Workers whose FIRST_NAME contains 'a'.
SELECT * FROM WORKER WHERE FIRST_NAME LIKE '%a%';

-- Q-17. Write an SQL query to print details of the Workers whose FIRST_NAME ends with 'a'.
SELECT * FROM WORKER WHERE FIRST_NAME LIKE '%a';

-- Q-18. Write an SQL query to print details of the Workers whose FIRST_NAME ends with 'h' and contains six alphabets.
SELECT * FROM WORKER WHERE FIRST_NAME LIKE '_____H';

-- Q-19. Write an SQL query to print details of the Workers whose SALARY lies between 100000 and 500000.
SELECT * FROM WORKER WHERE SALARY BETWEEN 100000 AND 500000;

-- Q-20. Write an SQL query to print details of the Workers who have joined in Feb'2014.
SELECT * FROM WORKER WHERE YEAR(JOINING_DATE)=2014 AND MONTH(JOINING_DATE)=02;

-- Q-21. Write an SQL query to fetch the count of employees working in the department 'Admin'.
SELECT DEPARTMENT ,COUNT(*) AS EMPLOYEE FROM WORKER WHERE DEPARTMENT ='ADMIN';
select department, count(*) from worker where department = 'Admin';

-- Q-22. Write an SQL query to fetch worker full names with salaries >= 50000 and <= 100000.
SELECT CONCAT(FIRST_NAME, ' ', LAST_NAME) AS FULL_NAME FROM WORKER WHERE SALARY >= 50000 AND SALARY <= 100000;

-- Q-23. Write an SQL query to fetch the no. of workers for each department in the descending order.
SELECT DEPARTMENT,COUNT(WORKER_ID) AS NO_OF_WORKER FROM WORKER GROUP BY DEPARTMENT ORDER BY NO_OF_WORKER DESC;

-- Q-24. Write an SQL query to print details of the Workers who are also Managers.
SELECT W.* FROM WORKER AS W INNER JOIN TITLE AS T ON W.WORKER_ID= T.WORKER_REF_ID WHERE T.WORKER_TITLE= 'MANAGER';
SELECT * FROM WORKER AS W INNER JOIN TITLE AS T ON W.WORKER_ID= T.WORKER_REF_ID WHERE T.WORKER_TITLE= 'MANAGER';

-- Q-25. Write an SQL query to fetch number (more than 1) of same titles in the ORG of different types.

```
SELECT WORKER_TITLE, COUNT(WORKER_REF_ID)AS TITLES  FROM TITLE  GROUP BY WORKER_TITLE HAVING TITLES >1;
```

-- Q-26. Write an SQL query to show only odd rows from a table.

```
SELECT * FROM WORKER WHERE MOD(WORKER_ID, 2) !=0;
SELECT * FROM WORKER WHERE MOD(WORKER_ID, 2) <> 0;
```

-- Q-27. Write an SQL query to show only even rows from a table.

```
SELECT * FROM WORKER WHERE MOD(WORKER_ID, 2) = 0;
```

-- Q-28. Write an SQL query to clone a new table from another table.

```
CREATE TABLE WORKER_CLONE LIKE WORKER;
INSERT INTO WORKER_CLONE SELECT * FROM WORKER;
SELECT * FROM WORKER_CLONE;
```

-- Q-29. Write an SQL query to fetch intersecting records of two tables. COOMAN DATA BETWEEN TWO TABLES

```
SELECT WORKER.* FROM WORKER INNER JOIN WORKER_CLONE USING(WORKER_ID);
```

-- Q-30. Write an SQL query to show records from one table that another table does not have.
-- MINUS LIKE (A- B)

```
SELECT WORKER.* FROM WORKER LEFT JOIN WORKER_CLONE USING(WORKER_ID)
WHERE WORKER_CLONE.WORKER_ID IS NULL;
```

-- Q-31. Write an SQL query to show the current date and time.
-- DUAL

```
SELECT NOW();
SELECT CURRENT_DATE();
```

-- Q-32. Write an SQL query to show the top n (say 5) records of a table order by descending salary.

```
SELECT * FROM WORKER ORDER BY SALARY DESC LIMIT 5;
```

-- Q-33. Write an SQL query to determine the nth (say n=5) highest salary from a table.

```
SELECT * FROM WORKER ORDER BY SALARY DESC LIMIT 4, 1;
SELECT DISTINCT SALARY FROM WORKER ORDER BY SALARY DESC LIMIT 4,1;
```

-- IF WE WANT THE 3RD HIGHEST SALARY FROM TABLE (2) IS WORK AS OFFSET LIKE AS FORMULA N-1, 1;

```
SELECT * FROM WORKER ORDER BY SALARY DESC LIMIT 1,1;
```

-- Q-34. Write an SQL query to determine the 5th highest salary without using LIMIT keyword.
-- CORELATED SUBQUERY USING TO SOLE THE QUES

```
SELECT * FROM WORKER AS W1
WHERE 4=(
SELECT COUNT(DISTINCT (W2.SALARY))
FROM WORKER W2
WHERE W2.SALARY >= W1.SALARY
);

SELECT * FROM WORKER ORDER BY SALARY DESC LIMIT 4, 1;
```

-- Q-35. Write an SQL query to fetch the list of employees with the same salary.

```
SELECT W.* FROM WORKER AS W ,WORKER_CLONE AS W1 WHERE W.SALARY =
W1.SALARY AND W.WORKER_ID != W1.WORKER_ID;
```

-- Q-36. Write an SQL query to show the second highest salary from a table using sub-query.

```
SELECT MAX(SALARY) FROM WORKER
WHERE SALARY NOT IN (SELECT MAX(SALARY) FROM WORKER);
```

-- BY USING LIMIT

```
SELECT DISTINCT SALARY FROM WORKER ORDER BY SALARY DESC LIMIT 1,1;
```

-- Q-37. Write an SQL query to show one row twice in results from a table.
-- UNION TO GIVE THE DISTINCT VALUE FROM TABLE AND UNION ALL TO GIVE THE SAME TABLE IN TWICE

```
SELECT * FROM WORKER
UNION ALL
SELECT * FROM WORKER ORDER BY WORKER_ID;
```

-- Q-38. Write an SQL query to list worker_id who does not get bonus.
-- use the subquery

```
SELECT Worker_id FROM WORKER AS W WHERE WORKER_ID NOT IN (SELECT
WORKER_REF_ID FROM BONUS);
```

-- Q-39. Write an SQL query to fetch the first 50% records from a table. 4 IS 50% PERCENT

```
SELECT * FROM WORKER WHERE WORKER_ID <= (SELECT COUNT(WORKER_ID)/2 FROM
WORKER);
```

-- Q-40. Write an SQL query to fetch the departments that have less than 4 people in it.

```
SELECT DEPARTMENT,  COUNT(DEPARTMENT)AS DEP FROM WORKER GROUP BY
DEPARTMENT HAVING DEP < 4;
```

-- Q-41. Write an SQL query to show all departments along with the number of people in there.

```
SELECT DEPARTMENT , COUNT(DEPARTMENT) AS NO_PEOPLE FROM  WORKER GROUP BY DEPARTMENT;
```

-- Q-42. Write an SQL query to show the last record from a table.

```
SELECT * FROM WORKER WHERE WORKER_ID=(SELECT MAX(WORKER_ID) FROM WORKER);
```

-- Q-43. Write an SQL query to fetch the first row of a table.

```
SELECT * FROM WORKER WHERE WORKER_ID=(SELECT MIN(WORKER_ID) FROM WORKER);
SELECT * FROM WORKER LIMIT 1;
```

-- Q-44. Write an SQL query to fetch the last five records from a table.

```
(SELECT * FROM WORKER ORDER BY WORKER_ID DESC LIMIT 5) ORDER BY WORKER_id;
```

-- Q-45. Write an SQL query to print the name of employees having the highest salary in each department.

```
SELECT W.DEPARTMENT, W.SALARY, W.FIRST_NAME FROM (SELECT MAX(SALARY) AS MAXSAL, DEPARTMENT FROM WORKER GROUP BY DEPARTMENT) TEMP
INNER JOIN WORKER AS W ON TEMP.DEPARTMENT =  W.DEPARTMENT AND TEMP.MAXSAL= W.SALARY;
```

-- Q-46. Write an SQL query to fetch three max salaries from a table using co-related subquery

```
SELECT DISTINCT(SALARY) FROM WORKER  ORDER BY SALARY DESC LIMIT 3;

SELECT DISTINCT SALARY FROM WORKER AS W1
WHERE 3>= (
SELECT COUNT(DISTINCT (W2.SALARY))
FROM WORKER AS W2
WHERE W2.SALARY >= W1.SALARY
)ORDER BY W1.SALARY DESC;
-- DRY RUN AFTER REVISING THE CORELATED SUBQUERY CONCEPT FROM LEC-9.
```

-- Q-47. Write an SQL query to fetch three min salaries from a table using co-related subquery

```
SELECT DISTINCT SALARY FROM WORKER AS W1
WHERE 3>= (
SELECT COUNT(DISTINCT (W2.SALARY))
FROM WORKER AS W2
WHERE W2.SALARY <= W1.SALARY
)ORDER BY W1.SALARY;

SELECT DISTINCT SALARY FROM WORKER ORDER BY SALARY LIMIT 3;
```

-- Q-48. Write an SQL query to fetch nth max salaries from a table.

```
SELECT DISTINCT SALARY FROM WORKER AS W1
WHERE N>= (
SELECT COUNT(DISTINCT SALARY)
FROM WORKER AS W2
WHERE W2.SALAEY >= W1.SALARY
)ORDER BY W1.SALARY;
```

-- Q-49. Write an SQL query to fetch departments along with the total salaries paid for each of them.

```
SELECT DEPARTMENT, SUM(SALARY) AS TOTAL_SALARY FROM WORKER GROUP BY
DEPARTMENT;
```

-- Q-50. Write an SQL query to fetch the names of workers who earn the highest salary.

```
SELECT FIRST_NAME , SALARY FROM WORKER WHERE SALARY=(SELECT MAX(SALARY)
FROM WORKER);
```

-- 51 QUESTION

```
CREATE TABLE PAIRS(
A INT,
B INT
);

INSERT INTO PAIRS VALUES (1,2),(2,4),(2,1),(3,2),(4,2),(5,6),(6,5),(7,8);
SELECT * FROM PAIRS;
```

-- TO REMOVE THE REVERSE ORDER OR DUPLICATE VALUES FROM THE DATA

-- USING JOINTS

```
SELECT LT.* FROM PAIRS AS LT LEFT JOIN PAIRS AS RT
ON LT.A = RT.B AND LT.B = RT.A
WHERE RT.A IS NULL OR LT.A < RT.A;
```

-- USING CORRELATED SUBQUERY

```
SELECT * FROM PAIRS AS LT
WHERE NOT EXISTS
(SELECT * FROM PAIRS AS RT WHERE LT.A= RT.B AND LT.B= RT.A AND LT.A > RT.A);
```