```
create database org;
USE org;
CREATE table Worker (
                    WORKER_ID INT NOT NULL PRIMARY KEY auto_increment,
                    FIRST_NAME CHAR(25),
                    LAST_NAME CHAR(25),
                    SALARY INT,
                    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;

CREATE TABLE Bonus (
                      WORKER_REF_ID INT,
                      BONUS_AMOUNT INT,
                      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');

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', '2016-02-20 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');
```

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

```sql
SELECT FIRST_NAME AS WORKER_NAME FROM Worker;
```

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

```sql
SELECT UPPER(FIRST_NAME) FROM Worker;
```

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

```sql
SELECT distinct department from worker;
```

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

```sql
select substring(first_name, 1, 3) from worker;
```

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

```sql
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 right side.

```sql
select rtrim(first_name) from worker;
```

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

```sql
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.

```sql
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’.

```sql
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.)

```sql
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.

```sql
select * from worker order by first_name;
```

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

```sql
select * from worker order by department desc, first_name;
```

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

```sql
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.

```sql
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 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 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(*) 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) from worker where salary between 50000 and 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 * from worker as emp inner join title as ttl on emp.worker_id = ttl.worker_ref_id
where ttl.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(*) as count from title group by worker_title having count > 1;
```

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

```
select * from worker where mod(worker_id, 2) != 0;
-- or
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.

```
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

```
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 curdate();
select curtime();
select now();
```

-- 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;
```

-- Q-34. Write an SQL query to determine the 5th highest salary without using LIMIT keyword.

```
select * from worker w1 where 4 = (select count(distinct w2.salary) from worker w2 where
w2.salary >=   w1.salary);
```

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

```
select * from worker w1, worker w2 where w1.salary = w2.salary and w1.worker_id != w2.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);
```

-- Q-37. Write an SQL query to show one row twice in results from a table.

```
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.

```
select w.* from worker w left join bonus b on w.worker_id = b.worker_ref_id where b.worker_ref_id is NULL;
 -- OR
select worker_id from worker 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.

```
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) cd from worker group by department having cd < 4;
```

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

```
select department, count(department) 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);
```

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

```
select * from worker where worker_id > (select count(worker_id)-5 from worker);
 -- OR
(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 concat(w.first_name, ' ', w.last_name), w.department, w.salary
from (select department, max(salary) sal from worker group by department) as temp
inner join worker w on w.department = temp.department and w.salary = temp.sal;
```

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

```
select distinct salary from worker w1 where 3 = (select count( distinct salary)
from worker w2 where w2.salary >= w1.salary) order by salary desc;
```

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

```
select distinct salary from worker w1 where 3 >= (select count(distinct salary) from worker w2 where
w2.salary <= w1.salary) order by salary;
```

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

```
select distinct salary from worker w1 where n = (select count( distinct salary) from worker w2 where
 w2.salary >= w1.salary) order by salary desc;
```

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

```
select department, sum(salary) as sum_sal from worker group by department order by sum_sal desc;
```

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

```
select concat(first_name, ' ', last_name) as emp_name, salary from worker where
salary = (select max(salary) from worker);
```

```sql
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;

-- Q-51. Remove Revesed Pairs
-- method1: using joins

    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 rt.A > rt.B;

-- method2: corelated subquery

   select * from pairs p1 where not exists (select * from pairs p2 where p1.A = p2.B and p1.B = p2.A and p1.A > p2.A);
```