**SQL Practical Questions**

1. Write a query to select all columns from the 'employees' table.  
 Answer: select \* from employees;

Explanation: Retrieved all the records from the table using select statement.

2. Write a query to select 'first\_name' and 'salary' from 'employees' where salary > 50000.

Answer: select first\_name, salary from employees where salary > 50000;

Explanation: We have use where condition to retrieve only the specific records according to the condition.

3. Retrieve all records from the 'departments' table where department\_name is 'Sales'.

Answer: select \* from departments where department\_name = ‘Sales’;

Explanation: We have use where condition to retrieve only the specific records according to the condition.

4. List all employees whose names start with 'A'.

Answer: select \* from employees where name like ‘A%’;

Explanation: We have used wildcards here to get the employees starting their names with A.

5. Display unique job titles from the 'jobs' table.

Answer: select distinct job\_title from jobs;

Explanation: We have use distinct keyword to get unique job titles in table.

6. Count how many employees are in each department.

Answer: select count(id) as employee\_count, department\_id from employee group by department;

Explanation: Used count for calculating the number of employees, group by for including each the department.

7. Find the average salary in each department.

Answer: select avg(salary) as avg\_salary, department\_id from employees group by department\_id;

Explanation: Used avg for calculating average salary.

8. List employees ordered by hire\_date descending.

Answer: select \* from employees order by hire\_date desc;

Explanation: Used order by desc for ordering in descending order.

9. Write a query to find employees with NULL commission\_pct.

Answer: select \* from employees where commission\_pct is null;

Explanation: Used is null statements

10. Retrieve all departments that have more than 5 employees.

Answer: select department\_id, count(id) as employee\_count from employees where count(id) > 5;

Explanation: Used count aggregation function

11. Write an INNER JOIN between 'employees' and 'departments' on department\_id.

Answer: select e.id,e.name, e.salary, d.department\_id from employees e inner join department d on e.department\_id = d.department\_id;

Explanation: Used inner join which retrieving only matching columns from both tables

12. Use LEFT JOIN to find employees even if they are not assigned to any department.

Answer: select e.id,e.name, e.salary, d.department\_name from employees e left join department d on e.department\_id = d.department\_id;

Explanation: Used left join which retrieving matching columns from right table and all columns from left table

13. List employee names along with their manager names using a self join.

Answer: select e.name as employee\_name , m.name as manager\_name from employees e join employees m on e.manager\_id = m.employee\_id;

Explanation: Used join keyword which retrieves data from same table

14. Write a query using RIGHT JOIN between 'jobs' and 'employees'.

Answer: select e.id, e.name, j.job\_titile, j.job\_id from jobs j right join employees e on j.job\_id = e.job\_id;

Explanation: Used right join which retrieving matching columns from left table and all columns from right table.

15. Perform a FULL OUTER JOIN between 'departments' and 'employees'. (simulate in MySQL)

16. Show total salary paid per job\_id.

Answer: select job\_id, sum(salary) as total\_salary from employees group by job\_id;

Explaination: Used sum aggregate function

17. Find the department with the maximum total salary.

Answer: select department\_id, sum(salary) as total\_salary from employees group by department\_id order by total\_salary desc limit 1;

Explaination: Used sum function, order by for sorting and desc limit to get maximum

18. Display departments with average salary above 60000.

Answer: select avg(salary) as average\_salary, department\_id from employees group by department\_id where avg(salary) > 60000;

Explanation: Used average aggregate function

19. Count number of employees in each job group.

20. List all job titles with their maximum and minimum salary.

Answer: select j.job\_title, e.min(salary) as min\_salary, e.max(salary) as max\_salary from jobs j join employees e on j.job\_id = e.job\_id group by job\_title;

Explanation: Joined employees and jobs table, used min and max functions

21. Find employees earning more than the average salary.

Answer: select \* from employees where salary > (select avg(salary) from employees);

Explanation: Used average aggregate function

22. List departments where the maximum salary is over 100000.

Answer: select department\_id, from employees group by department\_id having max(salary) > 100000;

Explanation: Used max aggregate function

23. Get names of employees who work in the same department as 'John'.

24. List employees who are not managers (no one reports to them).

25. Find employees hired before their managers.

26. Display employee names in uppercase.

Answer: select upper(name) as uppercase\_name form employees;

Explanation: Used upper function

27. Concatenate first\_name and last\_name as 'Full Name'.

Answer: select concat(first\_name, “ ”, last\_name) as Full\_Name from employees;

Explanation: Used concat function

28. Show the length of each employee's first name.

Answer: select name, len(first\_name) as name\_length from employees;

Explanation: Used len function

29. Extract the month from hire\_date.

Answer: select name, month(hire\_date) as hire\_month from employees;

Explanation: Used month function.

30. Show current system date and time.

31. Round all salaries to 2 decimal places.

Answer: select name, round(salary, 2) as new\_salary from employees;

Explanation: Used round function

32. Show salary difference between highest and lowest paid employee.

Answer: select max(salary),min(salary as salary\_difference from employees;

Explanation: Used min and max aggregate function

33. Display absolute difference in salaries between two employees.

34. Show all employees whose name contains the letter 'e'.

Answer: select \* employees where name like “%e%”;

Explanation: Used like, a wildcard

35. List employees with salary between 50000 and 80000.

Answer: select \* from employees where salary between 50000 and 80000;

Explanation: Used between as we have find range

36. Create a table named 'projects' with columns: id, name, start\_date.

Answer: create table projects (id int primary key, name varchar(100),start\_date date);

Explanation: Used create table operation

37. Add a NOT NULL constraint to 'start\_date' in 'projects'.

38. Insert a record into 'projects' table.

Answer: insert into projects values (1, ‘Web Development’, ‘2022-12-03’);

Explanation: Used insert into to insert data

39. Delete a project with id = 3 from 'projects'.

Answer: delete from projects where id=3;

Explanation: Used delete from statment

40. Alter 'projects' to add column 'status' with default 'active'.

Answer: alter table projects add column status varchar(20) default 'active’;

Explanation: Used alter table statement

41. Start a transaction, update an employee's salary, then rollback.

Answer: start transaction;  
 update employees set salary = ‘60000’ where id = 3;

rollback;

Explanation: Used update statment

42. Use COMMIT to save changes after updating the 'departments' table.

43. Explain a use case for SAVEPOINT and demonstrate it with queries.

44. Create a view for employees with salary > 70000.

Answer: create view info as select \* from employees where salary > 70000;

Explanation: Used create view statement

45. Drop the view 'high\_salary\_employees'.

Answer: drop view high\_salary\_employees;

Explanation: Used drop view statement

46. Create an index on 'last\_name' in 'employees' table.

Answer: create index idx on employees(last\_name);

Explanation: Used create index statment

47. Drop the index from 'employees' table.

Answer: drop index idx on employees;

Explanation: Used drop index statement

48. Use ROW\_NUMBER() to rank employees by salary within each department.

49. Use RANK() to identify top 3 earners per department.

50. Use DENSE\_RANK() for same salary employees.