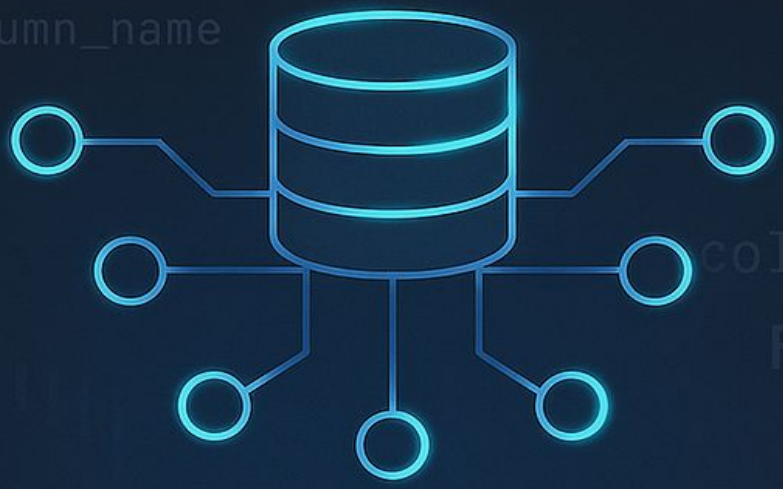

100 ESSENTIAL SQL QUERIES

Questions & Answers | Complete Reference Guide

Contents

1 Basic SQL Query Questions

1. **Select all records from a table.**

```sql
SELECT * FROM employees;
```

2. **Select specific columns.**

```sql
SELECT first_name, salary FROM employees;
```

3. **Filter rows using WHERE.**

```sql
SELECT * FROM employees WHERE salary > 50000;
```

4. **Use AND/OR operators.**

```sql
SELECT * FROM employees
WHERE department = 'IT' AND salary > 60000;
```

5. **Use BETWEEN operator.**

```sql
SELECT * FROM employees
WHERE salary BETWEEN 40000 AND 80000;
```

6. **Use LIKE operator.**

```sql
SELECT * FROM employees WHERE first_name LIKE 'A%';
```

7. **Use IN operator.**

```sql
SELECT * FROM employees
WHERE department IN ('IT', 'HR');
```

8. **Sort results using ORDER BY.**

```sql
SELECT * FROM employees ORDER BY salary DESC;
```

9. **Count total rows in a table.**

```sql
SELECT COUNT(*) FROM employees;
```

10. **Find distinct values.**

```sql
SELECT DISTINCT department FROM employees;
```

2 Aggregate Functions

11. **Find average salary.**

```sql
SELECT AVG(salary) FROM employees;
```

12. **Find maximum salary.**

```sql
SELECT MAX(salary) FROM employees;
```

13. **Find minimum salary.**

```sql
SELECT MIN(salary) FROM employees;
```

14. **Find sum of salaries.**

```sql
SELECT SUM(salary) FROM employees;
```

15. **Group by department.**

```sql
SELECT department, COUNT(*)
FROM employees
GROUP BY department;
```

16. **Group by and filter using HAVING.**

```sql
SELECT department, AVG(salary)
FROM employees
GROUP BY department
HAVING AVG(salary) > 60000;
```

17. **Get highest-paid employee per department.**

```sql
SELECT department, MAX(salary)
FROM employees
GROUP BY department;
```

18. **Count employees earning more than 70K.**

```sql
SELECT COUNT(*) FROM employees WHERE salary > 70000;
```

19. **Get department with minimum avg salary.**

```sql
SELECT TOP 1 department, AVG(salary) AS avg_sal
FROM employees
GROUP BY department
ORDER BY avg_sal ASC;
```

20. **Count employees per department.**

```sql
SELECT department, COUNT(*)
FROM employees
GROUP BY department;
```

3 Joins

21. **Inner Join two tables.**

```sql
SELECT e.first_name, d.department_name
FROM employees e
INNER JOIN departments d ON e.department_id = d.department_id;
```

22. **Left Join example.**

```sql
SELECT e.first_name, d.department_name
FROM employees e
LEFT JOIN departments d ON e.department_id = d.department_id;
```

23. **Right Join example.**

```sql
SELECT e.first_name, d.department_name
FROM employees e
RIGHT JOIN departments d ON e.department_id = d.department_id;
```

24. **Full Join example.**

```sql
SELECT e.first_name, d.department_name
FROM employees e
FULL OUTER JOIN departments d ON e.department_id = d.department_id;
```

25. **Join more than two tables.**

```sql
SELECT e.first_name, d.department_name, l.city
FROM employees e
JOIN departments d ON e.department_id = d.department_id
JOIN locations l ON d.location_id = l.location_id;
```

26. **Find employees without departments.**

```sql
SELECT e.*
FROM employees e
LEFT JOIN departments d ON e.department_id = d.department_id
WHERE d.department_id IS NULL;
```

27. **Find departments with no employees.**

```sql
SELECT d.*
FROM departments d
LEFT JOIN employees e ON d.department_id = e.department_id
WHERE e.employee_id IS NULL;
```

28. **Join and use aggregate.**

```sql
SELECT d.department_name, COUNT(e.employee_id)
FROM employees e
JOIN departments d ON e.department_id = d.department_id
GROUP BY d.department_name;
```

29. **Self Join example.**

```sql
SELECT e.first_name AS Employee, m.first_name AS Manager
FROM employees e
JOIN employees m ON e.manager_id = m.employee_id;
```

30. **Cross Join example.**

```sql
SELECT e.first_name, d.department_name
FROM employees e, departments d;
```

4 Subqueries

31. **Find employees earning above average salary.**

```sql
SELECT first_name, salary
FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees);
```

32. **Find second-highest salary.**

```sql
SELECT MAX(salary) FROM employees
WHERE salary < (SELECT MAX(salary) FROM employees);
```

33. **Find employees in same department as 'John'.**

```sql
SELECT *
FROM employees
WHERE department_id = (SELECT department_id FROM employees WHERE
    first_name='John');
```

34. **Correlated subquery example.**

```sql
SELECT e.first_name, e.salary
FROM employees e
WHERE e.salary > (SELECT AVG(salary) FROM employees WHERE department_id
    = e.department_id);
```

35. **Exists example.**

```sql
SELECT department_name
FROM departments d
WHERE EXISTS (SELECT * FROM employees e WHERE e.department_id = d.
    department_id);
```

36. **Not Exists example.**

```sql
SELECT department_name
FROM departments d
WHERE NOT EXISTS (SELECT * FROM employees e WHERE e.department_id = d.
    department_id);
```

37. **IN subquery example.**

```sql
SELECT *
FROM employees
WHERE department_id IN (SELECT department_id FROM departments WHERE
    location_id = 1700);
```

38. **Subquery in SELECT clause.**

```sql
SELECT first_name, (SELECT department_name FROM departments WHERE
    department_id = e.department_id) AS dept
FROM employees e;
```

39. **Subquery in FROM clause.**

```sql
SELECT dept, AVG(sal)
FROM (SELECT department_id AS dept, salary AS sal FROM employees) t
GROUP BY dept;
```

40. **Find top 3 salaries.**

```sql
SELECT DISTINCT salary
FROM employees e1
WHERE 3 > (SELECT COUNT(DISTINCT salary) FROM employees e2 WHERE e2.
    salary > e1.salary);
```

5 Advanced Query Practice

41. **Rank employees by salary.**

```sql
SELECT first_name, salary, RANK() OVER (ORDER BY salary DESC) AS Rank
    FROM employees;
```

42. **Dense Rank example.**

```sql
SELECT first_name, salary, DENSE_RANK() OVER (ORDER BY salary DESC) AS
    Rank FROM employees;
```

43. **Find nth highest salary using CTE.**

```sql
WITH salary_cte AS (
 SELECT salary, DENSE_RANK() OVER (ORDER BY salary DESC) AS rnk
 FROM employees
)
SELECT salary FROM salary_cte WHERE rnk = 3;
```

44. **Find duplicate records.**

```sql
SELECT first_name, COUNT(*)
FROM employees
GROUP BY first_name
HAVING COUNT(*) > 1;
```

45. **Delete duplicate rows.**

```sql
DELETE FROM employees
WHERE employee_id NOT IN (
 SELECT MIN(employee_id) FROM employees GROUP BY first_name, last_name
);
```

46. **Find top 5 highest salaries.**

```sql
SELECT TOP 5 * FROM employees ORDER BY salary DESC;
```

47. **Find top N records per department.**

```sql
SELECT * FROM (
 SELECT e.*, ROW_NUMBER() OVER (PARTITION BY department_id ORDER BY
     salary DESC) AS rn
 FROM employees e
) t
WHERE rn <= 3;
```

48. **Find employees who joined in 2022.**

```sql
SELECT * FROM employees WHERE YEAR(hire_date) = 2022;
```

49. **Find employees whose name starts and ends with same letter.**

```sql
SELECT * FROM employees WHERE LEFT(first_name,1) = RIGHT(first_name,1);
```

50. **Find employees with NULL commission.**

```sql
SELECT * FROM employees WHERE commission_pct IS NULL;
```

6 String & Date Functions

51. **Convert names to uppercase.**

```sql
SELECT UPPER(first_name) FROM employees;
```

52. **Concatenate first and last name.**

```sql
SELECT CONCAT(first_name, ' ', last_name) AS full_name FROM employees;
```

53. **Find string length.**

```sql
SELECT LEN(first_name) FROM employees;
```

54. **Substring example.**

```sql
SELECT SUBSTRING(first_name, 1, 3) FROM employees;
```

55. **Replace substring.**

```sql
SELECT REPLACE(first_name, 'a', 'A') FROM employees;
```

56. **Current date.**

```sql
SELECT GETDATE();
```

57. **Find employees hired in last 30 days.**

```sql
SELECT * FROM employees WHERE hire_date >= DATEADD(DAY, -30, GETDATE());
```

58. **Difference in years between two dates.**

```sql
SELECT DATEDIFF(YEAR, hire_date, GETDATE()) AS Experience FROM employees
    ;
```

59. **Add 10 days to date.**

```sql
SELECT DATEADD(DAY, 10, hire_date) FROM employees;
```

60. **Format date.**

```sql
SELECT FORMAT(hire_date, 'dd-MMM-yyyy') FROM employees;
```

7 DDL / DML Related Queries

61. **Create table.**

```sql
CREATE TABLE students (id INT, name VARCHAR(50), age INT);
```

62. **Insert record.**

```sql
INSERT INTO students VALUES (1, 'Rahul', 22);
```

63. **Update record.**

```sql
UPDATE students SET age = 23 WHERE id = 1;
```

64. **Delete record.**

```sql
DELETE FROM students WHERE id = 1;
```

65. **Add new column.**

```sql
ALTER TABLE students ADD city VARCHAR(50);
```

66. **Drop column.**

```sql
ALTER TABLE students DROP COLUMN city;
```

67. **Rename table.**

```sql
EXEC sp_rename 'students', 'pupils';
```

68. **Truncate table.**

```sql
TRUNCATE TABLE students;
```

69. **Drop table.**

```sql
DROP TABLE students;
```

70. **Create a view.**

```sql
CREATE VIEW high_salary AS SELECT * FROM employees WHERE salary > 80000;
```

8 Constraints & Keys

71. **Primary key example.**

```sql
CREATE TABLE dept (id INT PRIMARY KEY, name VARCHAR(50));
```

72. **Foreign key example.**

```sql
CREATE TABLE emp (id INT PRIMARY KEY, dept_id INT REFERENCES dept(id));
```

73. **Unique constraint.**

```sql
ALTER TABLE employees ADD CONSTRAINT uq_email UNIQUE (email);
```

74. **Check constraint.**

```sql
ALTER TABLE employees ADD CONSTRAINT chk_salary CHECK (salary > 0);
```

75. **Default constraint.**

```sql
ALTER TABLE employees ADD CONSTRAINT def_city DEFAULT 'Delhi' FOR city;
```

76. **Not null constraint.**

```sql
ALTER TABLE employees ALTER COLUMN first_name VARCHAR(50) NOT NULL;
```

77. **Composite key example.**

```sql
CREATE TABLE orders (order_id INT, product_id INT, PRIMARY KEY(order_id,
    product_id));
```

78. **Find all constraints.**

```sql
SELECT * FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS;
```

79. **Drop constraint.**

```sql
ALTER TABLE employees DROP CONSTRAINT chk_salary;
```

80. **Add foreign key after table creation.**

```sql
ALTER TABLE emp ADD FOREIGN KEY (dept_id) REFERENCES dept(id);
```

9 Indexes & Performance

81. **Create index.**

```sql
CREATE INDEX idx_salary ON employees(salary);
```

82. **Drop index.**

```sql
DROP INDEX idx_salary ON employees;
```

83. **Composite index.**

```sql
CREATE INDEX idx_name_dept ON employees(first_name, department_id);
```

84. **View indexes.**

```sql
EXEC sp_helpindex 'employees';
```

85. **Why use index?**

> Speeds up SELECT queries but slows down INSERT/UPDATE/DELETE.

86. **Clustered vs Non-clustered index difference.**

> - Clustered: Sorts and stores data physically.
> - Non-clustered: Creates a separate structure referencing data.

87. **Find slow queries.**

```sql
SELECT * FROM sys.dm_exec_query_stats;
```

88. **Rebuild index.**

```sql
ALTER INDEX ALL ON employees REBUILD;
```

89. **Disable index.**

```sql
ALTER INDEX idx_salary ON employees DISABLE;
```

90. **Enable index.**

```sql
ALTER INDEX idx_salary ON employees REBUILD;
```

10 Miscellaneous & Advanced

91. **Case statement.**

```sql
SELECT first_name,
 CASE WHEN salary > 80000 THEN 'High'
 WHEN salary BETWEEN 50000 AND 80000 THEN 'Medium'
 ELSE 'Low' END AS salary_range
FROM employees;
```

92. **COALESCE example.**

```sql
SELECT first_name, COALESCE(phone, 'Not Provided') FROM employees;
```

93. **ISNULL example.**

```sql
SELECT ISNULL(commission_pct, 0) FROM employees;
```

94. **Find employees hired after their manager.**

```sql
SELECT e.first_name, m.first_name AS manager
FROM employees e JOIN employees m ON e.manager_id = m.employee_id
WHERE e.hire_date < m.hire_date;
```

95. **Pivot example.**

```sql
SELECT * FROM
(SELECT department, gender, salary FROM employees) src
PIVOT (AVG(salary) FOR gender IN ([Male],[Female])) pvt;
```

96. **Unpivot example.**

```sql
SELECT * FROM salaries
UNPIVOT (value FOR type IN (basic, hra, bonus)) AS unpvt;
```

97. **CTE to count employees.**

```sql
WITH emp_cte AS (SELECT department_id, COUNT(*) AS total FROM employees
    GROUP BY department_id)
SELECT * FROM emp_cte WHERE total > 5;
```

98. **Recursive CTE (manager hierarchy).**

```sql
WITH emp_hierarchy AS (
 SELECT employee_id, manager_id, first_name FROM employees WHERE
     manager_id IS NULL
 UNION ALL
 SELECT e.employee_id, e.manager_id, e.first_name
 FROM employees e JOIN emp_hierarchy h ON e.manager_id = h.employee_id
)
SELECT * FROM emp_hierarchy;
```

99. **Find 3rd maximum salary using ROW_NUMBER.**

```sql
SELECT salary FROM (
 SELECT salary, ROW_NUMBER() OVER (ORDER BY salary DESC) AS rn FROM
     employees
) t WHERE rn = 3;
```

100. **Display employee count by joining month.**

```sql
SELECT MONTH(hire_date) AS join_month, COUNT(*)
FROM employees
GROUP BY MONTH(hire_date);
```