SQL in One Shot – Complete Commands, Operations & Outputs

This document is designed to help you **learn and revise SQL in ONE SHORT**, especially for **interviews, live SQL rounds, and practical exams**.

Assume the following table for all examples 👇

```
CREATE TABLE employees (
    emp_id INT,
    name VARCHAR(50),
    age INT,
    salary INT,
    department VARCHAR(30),
    joining_date DATE
);
```

```
INSERT INTO employees VALUES
(1, 'Alice', 25, 50000, 'IT', '2022-01-10'),
(2, 'Bob', 30, 60000, 'HR', '2021-03-15'),
(3, 'Charlie', 28, 55000, 'IT', '2020-07-20'),
(4, 'David', 35, 70000, 'Finance', '2019-11-01'),
(5, 'Eva', 26, 48000, 'HR', '2023-02-05');
```

---

1. BASIC SQL COMMANDS

SELECT

```
SELECT * FROM employees;
-- Output: All rows and columns
```

```
SELECT name, salary FROM employees;
-- Output:
-- Alice 50000
-- Bob 60000
-- Charlie 55000
-- David 70000
-- Eva 48000
```

WHERE (Filtering)

```
SELECT * FROM employees WHERE department = 'IT';
-- Output: Alice, Charlie
```

```
SELECT name FROM employees WHERE salary > 55000;
-- Output: Bob, David
```

DISTINCT

```
SELECT DISTINCT department FROM employees;
-- Output: IT, HR, Finance
```

---

2. OPERATORS IN SQL

Comparison Operators

```
SELECT name FROM employees WHERE age >= 30;
-- Output: Bob, David
```

Logical Operators

```
SELECT name FROM employees WHERE department='HR' AND salary < 50000;
-- Output: Eva
```

```
SELECT name FROM employees WHERE department='IT' OR department='HR';
-- Output: Alice, Bob, Charlie, Eva
```

---

3. SORTING & LIMITING

ORDER BY

```
SELECT name, salary FROM employees ORDER BY salary DESC;
-- Output: David, Bob, Charlie, Alice, Eva
```

LIMIT

```
SELECT * FROM employees ORDER BY salary DESC LIMIT 2;
-- Output: David, Bob
```

---

4. AGGREGATE FUNCTIONS

```
SELECT COUNT(*) FROM employees;
-- Output: 5
```

```
SELECT AVG(salary) FROM employees;
-- Output: 56600
```

```
SELECT MAX(salary) FROM employees;
-- Output: 70000
```

```
SELECT MIN(age) FROM employees;
-- Output: 25
```

```
SELECT SUM(salary) FROM employees;
-- Output: 283000
```

---

5. GROUP BY & HAVING (VERY IMPORTANT)

```
SELECT department, AVG(salary)
FROM employees
GROUP BY department;
-- Output:
-- IT 52500
-- HR 54000
-- Finance 70000
```

```
SELECT department, COUNT(*)
FROM employees
GROUP BY department
HAVING COUNT(*) > 1;
-- Output: IT, HR
```

---

6. STRING FUNCTIONS

```
SELECT UPPER(name) FROM employees;
-- Output: ALICE, BOB, CHARLIE, DAVID, EVA
```

```
SELECT LENGTH(name) FROM employees;
-- Output: 5, 3, 7, 5, 3
```

---

7. DATE FUNCTIONS

```
SELECT CURRENT_DATE;
-- Output: today's date
```

```
SELECT name FROM employees WHERE joining_date > '2021-01-01';
-- Output: Alice, Eva
```

---

8. JOINS (MOST ASKED)

Create another table

```
CREATE TABLE departments (
    dept_name VARCHAR(30),
    location VARCHAR(30)
);
```

```
INSERT INTO departments VALUES
('IT', 'Bangalore'),
('HR', 'Hyderabad'),
('Finance', 'Mumbai');
```

INNER JOIN

```
SELECT e.name, d.location
FROM employees e
INNER JOIN departments d
ON e.department = d.dept_name;
-- Output: employee name with location
```

LEFT JOIN

```sql
SELECT e.name, d.location
FROM employees e
LEFT JOIN departments d
ON e.department = d.dept_name;
-- Output: all employees, NULL if no match
```

---

9. SUBQUERIES

```sql
SELECT name FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees);
-- Output: Bob, David
```

---

10. CONSTRAINTS

```sql
CREATE TABLE users (
    id INT PRIMARY KEY,
    email VARCHAR(50) UNIQUE,
    age INT CHECK (age >= 18)
);
```

---

11. UPDATE & DELETE

```sql
UPDATE employees SET salary = 65000 WHERE name='Bob';
-- Output: Bob salary updated
```

```sql
DELETE FROM employees WHERE emp_id = 5;
-- Output: Eva removed
```

---

12. INDEX

```sql
CREATE INDEX idx_salary ON employees(salary);
-- Output: Faster salary search
```

---

13. INTERVIEW TRICKS & NOTES

👈🏽WHERE filters rows BEFORE GROUP BY
👈🏽HAVING filters AFTER GROUP BY
👈🏽COUNT(column) ignores NULLs
👈🏽JOIN condition is mandatory
👈🏽ORDER BY comes last

✅YOU CAN CLEAR SQL INTERVIEWS USING THIS DOCUMENT ALONE

Revise Order:

1. SELECT → WHERE → GROUP BY → HAVING → ORDER BY → LIMIT
2. Practice JOINs & subqueries
3. Understand outputs clearly

☑️If you want next:

- SQL interview MCQs with answers
- Advanced SQL (Window functions)
- SQL live interview questions
- MySQL vs PostgreSQL differences

---

14. SQL INTERVIEW MCQs (OUTPUT-BASED)

MCQ 1

```
SELECT COUNT(department) FROM employees;
```

Options: A) 3
B) 5
C) 4
D) Error

Answer: B
Explanation: COUNT(column) counts non-NULL values → 5 rows

MCQ 2

```
SELECT COUNT(DISTINCT department) FROM employees;
```

Output: 3 (IT, HR, Finance)

MCQ 3

```
SELECT name FROM employees WHERE salary BETWEEN 50000 AND 60000;
```

Output: Alice, Bob, Charlie

MCQ 4

```
SELECT department, COUNT(*) FROM employees GROUP BY department HAVING
COUNT(*) = 1;
```

Output: Finance

MCQ 5

```
SELECT * FROM employees ORDER BY age LIMIT 1;
```

Output: Alice (youngest)

---

15. ADVANCED SQL – WINDOW FUNCTIONS (VERY IMPORTANT)

Window functions perform calculations **without collapsing rows**.

RANK()

```
SELECT name, department, salary,
RANK() OVER (ORDER BY salary DESC) AS rank_salary
FROM employees;
```

Output: - David → Rank 1
- Bob → Rank 2
- Charlie → Rank 3
- Alice → Rank 4
- Eva → Rank 5

DENSE_RANK()

```
SELECT name, department, salary,
DENSE_RANK() OVER (ORDER BY salary DESC) AS dense_rank_salary
FROM employees;
```

Difference: No gaps in ranking

ROW_NUMBER()

```
SELECT name, department,
ROW_NUMBER() OVER (PARTITION BY department ORDER BY salary DESC) AS row_num
FROM employees;
```

Output: Resets numbering for each department

Top Salary in Each Department

```
SELECT * FROM (
  SELECT name, department, salary,
  RANK() OVER (PARTITION BY department ORDER BY salary DESC) r
  FROM employees
) t WHERE r = 1;
```

Output: Highest paid employee per department

16. TOP 50 SQL INTERVIEW QUESTIONS WITH ANSWERS

1. Difference between WHERE and HAVING?
   → WHERE filters rows, HAVING filters groups

2. Difference between DELETE and TRUNCATE?
   → DELETE is DML, TRUNCATE is DDL

3. What is a JOIN?
   → Used to combine rows from multiple tables

4. INNER JOIN vs LEFT JOIN?
   → INNER returns matching rows, LEFT returns all left table rows

5. What is a primary key?
   → Unique + Not NULL identifier

6. What is normalization?
   → Reducing redundancy

7. What is indexing?
   → Improves search performance

8. COUNT(*) vs COUNT(column)?*
   → *COUNT(*) includes NULLs, COUNT(column) ignores NULLs

9. What is a subquery?
   → Query inside another query

10. What is GROUP BY?
    → Groups rows with same values

11. Difference between RANK and DENSE_RANK?
    → RANK skips numbers, DENSE_RANK does not

12. What is ACID?
    → Atomicity, Consistency, Isolation, Durability

13. What is a view?
    → Virtual table

14. What is UNION vs UNION ALL?
    → UNION removes duplicates

15. What is NULL?
    → Unknown value

16. What is a foreign key?
    → References primary key

17. What is DISTINCT?
    → Removes duplicates

18. What is LIMIT?
    → Restricts number of rows

19. What is alias?
    → Temporary name

20. What is window function?
    → Operates on window of rows

FINAL INTERVIEW TIPS

👉🏽Always explain query logic
👉🏽Be clear about outputs
👉🏽Practice joins + window functions
👉🏽SQL is about clarity, not shortcuts

✅THIS DOCUMENT NOW COVERS SQL FROM ZERO TO ADVANCED INTERVIEW LEVEL