---

# 📌 SQL Cheat Sheet (MySQL)

---

## 1️⃣ Basic SELECT

```
-- All columns
SELECT * FROM employees;

-- Specific columns
SELECT first_name, last_name FROM employees;

-- Unique values
SELECT DISTINCT department FROM employees;

-- Sorting
SELECT * FROM employees ORDER BY salary DESC, first_name ASC;

-- Limit results
SELECT * FROM employees LIMIT 5;
```

---

## 2️⃣ Filtering Data (WHERE)

```
-- Basic filters
SELECT * FROM employees WHERE salary > 50000;

-- Multiple conditions
SELECT * FROM employees
WHERE department = 'IT' AND salary > 60000;

-- Range
SELECT * FROM products WHERE price BETWEEN 50 AND 200;

-- List of values
SELECT * FROM customers WHERE city IN ('New York', 'LA');

-- Pattern matching
SELECT * FROM employees WHERE name LIKE 'A%';   -- Starts with A
SELECT * FROM employees WHERE name LIKE '%son'; -- Ends with son
```

---

## 3 Aggregate Functions

```
SELECT COUNT(*) AS total_employees FROM employees;
SELECT AVG(salary) AS avg_salary FROM employees;
SELECT SUM(amount) AS total_sales FROM orders;
SELECT MIN(price) AS cheapest FROM products;
SELECT MAX(price) AS most_expensive FROM products;
```

---

## 4 GROUP BY & HAVING

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

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

---

## 5 Joins

```
-- INNER JOIN
SELECT e.name, d.department_name
FROM employees e
INNER JOIN departments d ON e.department_id = d.id;

-- LEFT JOIN
SELECT c.name, o.order_id
FROM customers c
LEFT JOIN orders o ON c.id = o.customer_id;

-- RIGHT JOIN
SELECT o.order_id, c.name
FROM customers c
RIGHT JOIN orders o ON c.id = o.customer_id;

-- Multiple Joins
SELECT o.order_id, c.name, p.product_name
FROM orders o
JOIN customers c ON o.customer_id = c.id
JOIN products p ON o.product_id = p.id;
```

---

## 6 Subqueries

```
-- Single value
SELECT * FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees);

-- Multiple values
SELECT * FROM employees
WHERE department_id IN (
    SELECT department_id FROM departments WHERE location = 'NY'
);

-- Correlated
SELECT name FROM customers c
WHERE EXISTS (
    SELECT 1 FROM orders o WHERE o.customer_id = c.id
);
```

---

## 7 Conditional Logic

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

---

## 8 String Functions

```
SELECT CONCAT(first_name, ' ', last_name) AS full_name FROM employees;
SELECT UPPER(name) FROM customers;
SELECT LOWER(city) FROM customers;
SELECT SUBSTRING(name, 1, 3) FROM employees;
SELECT TRIM('  spaces  ') AS trimmed;
SELECT REPLACE(name, 'Corp', 'Corporation') FROM companies;
```

---

## 9 Date Functions

```
SELECT CURDATE();        -- Current date
```

```
SELECT NOW();            -- Date + Time
SELECT YEAR(NOW());      -- Current year
SELECT MONTH(NOW());     -- Current month
SELECT DATEDIFF(CURDATE(), hire_date) AS days_worked FROM employees;
SELECT * FROM orders WHERE YEAR(order_date) = 2024;
```

---

## 🔟 Window Functions (MySQL 8+)

```
-- Row number
SELECT name, salary, ROW_NUMBER() OVER(ORDER BY salary DESC) AS row_num
FROM employees;

-- Rank in department
SELECT department, name, RANK() OVER(PARTITION BY department ORDER BY salary
DESC) AS rank_in_dept
FROM employees;
```

---

## 💡 Interview-Tricky Queries

```
-- Second highest salary
SELECT MAX(salary) AS second_highest
FROM employees
WHERE salary < (SELECT MAX(salary) FROM employees);

-- Products never ordered
SELECT p.*
FROM products p
LEFT JOIN orders o ON p.id = o.product_id
WHERE o.product_id IS NULL;

-- Top 3 salaries per department
SELECT department, name, salary
FROM (
   SELECT department, name, salary,
          DENSE_RANK() OVER(PARTITION BY department ORDER BY salary DESC) AS rnk
   FROM employees
) t
WHERE rnk <= 3;
```

---

## 📌 Quick Syntax Recap Table

| Feature | Syntax Example |
| --- | --- |
| Select all | `SELECT * FROM table;` |
| Filtering | `WHERE col > 100` |
| Sorting | `ORDER BY col DESC` |
| Aggregates | `COUNT(), SUM(), AVG()` |
| Group & Filter | `GROUP BY col HAVING condition` |
| Inner Join | `JOIN ... ON ...` |
| Left Join | `LEFT JOIN ... ON ...` |
| Subquery | `WHERE col IN (SELECT col...)` |
| Case Condition | `CASE WHEN ... THEN ... END` |
| String Ops | `CONCAT(), UPPER(), LOWER()` |
| Date Ops | `NOW(), DATEDIFF()` |

---