# Ultimate SQL Cheat Sheet

**Learn SQL Basics to Advanced in One Go 🚀**

Keep this reference handy to quickly write queries, practice, and master SQL.

---

## 1. SQL Basics

```sql
-- Select all columns
SELECT * FROM employees;

-- Select specific columns
SELECT name, salary FROM employees;

-- Filtering rows
SELECT * FROM employees WHERE salary > 50000;

-- Sorting
SELECT * FROM employees ORDER BY salary DESC;

-- Limiting
SELECT * FROM employees LIMIT 5;
```



---

## 2. Joins

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

-- LEFT JOIN
SELECT e.name, d.department_name
FROM employees e
LEFT JOIN departments d ON e.dept_id = d.id;

-- RIGHT JOIN
SELECT e.name, d.department_name
FROM employees e
RIGHT JOIN departments d ON e.dept_id = d.id;

-- FULL OUTER JOIN
SELECT e.name, d.department_name
FROM employees e
FULL OUTER JOIN departments d ON e.dept_id = d.id;
```

## 3. Aggregations

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

-- Average salary per department
SELECT dept_id, AVG(salary)
FROM employees
GROUP BY dept_id;

-- Total sales by product
SELECT product_id, SUM(amount)
FROM sales
```

## 4. Window Functions

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

-- Running total
SELECT order_id, amount,
       SUM(amount) OVER (ORDER BY order_date) AS running_total
FROM orders;

-- Ranking
SELECT customer_id, SUM(total) AS total_spent,
       RANK() OVER (ORDER BY SUM(total) DESC) AS rank
FROM orders
GROUP BY customer_id;
```

## 5. Advanced SQL

```sql
-- Common Table Expression (CTE)
WITH top_sales AS (
    SELECT product_id, SUM(amount) AS total_sales
    FROM sales
    GROUP BY product_id
)
SELECT * FROM top_sales
ORDER BY total_sales DESC
LIMIT 5;

-- Recursive CTE (hierarchies)
WITH RECURSIVE employee_hierarchy AS (
    SELECT id, name, manager_id, 1 AS level
    FROM employees WHERE manager_id IS NULL
    UNION ALL
```

## 6. Optimization Tips

- ✅ Always use `EXPLAIN` to analyze queries.
- ✅ Use proper indexes for frequently searched columns.
- ✅ Avoid `SELECT *` in production.
- ✅ Break down complex queries with CTEs.
- ✅ Limit result sets for performance.

## 7. Bonus: Quick Reference Table

| Clause | Purpose |
| --- | --- |
| SELECT | Specifies columns to return |
| FROM | Specifies table to query |
| WHERE | Filters rows |
| GROUP BY | Groups rows by column values |
| HAVING | Filters groups |
| ORDER BY | Sorts results |