SQL Syntax Cheat Sheet

1. Basic SQL Syntax

```
-- SELECT: Used to retrieve data from a table
SELECT name, age FROM users;
SELECT * FROM products; -- selects all columns

-- INSERT: Adds new data into a table
INSERT INTO users (name, age) VALUES ('Alice', 25);

-- UPDATE: Modifies existing data
UPDATE users SET age = 26 WHERE name = 'Alice';

-- DELETE: Removes data from a table
DELETE FROM users WHERE age < 18;

-- CREATE TABLE: Defines a new table and its columns
CREATE TABLE employees (
  id INT PRIMARY KEY,
  name VARCHAR(100),
  position VARCHAR(50)
);

-- DROP TABLE: Deletes a table and all its data
DROP TABLE old_records;
```

2. Joins

```
-- INNER JOIN: Returns only matching rows from both tables
SELECT orders.id, customers.name
FROM orders
INNER JOIN customers ON orders.customer_id = customers.id;

-- LEFT JOIN: Returns all rows from the left table, even if no match in the right
SELECT customers.name, orders.id
FROM customers
LEFT JOIN orders ON customers.id = orders.customer_id;

-- RIGHT JOIN: Returns all rows from the right table, even if no match in the left
SELECT orders.id, customers.name
FROM customers
RIGHT JOIN orders ON customers.id = orders.customer_id;

-- FULL OUTER JOIN: Returns all rows when there is a match in either table
SELECT a.id, b.id
FROM tableA a
FULL OUTER JOIN tableB b ON a.id = b.id;
```

SQL Syntax Cheat Sheet

3. Filtering & Sorting

```
-- WHERE: Filters rows that meet the condition
SELECT * FROM products WHERE price > 100;

-- Logical Operators
SELECT * FROM products WHERE price > 100 AND stock > 0;

-- BETWEEN: Filters values in a range
SELECT * FROM events WHERE event_date BETWEEN '2024-01-01' AND '2024-12-31';

-- IN: Filters values from a list
SELECT * FROM users WHERE country IN ('USA', 'Canada');

-- LIKE: Pattern matching (use % as wildcard)
SELECT * FROM users WHERE name LIKE 'J%';

-- IS NULL: Checks for missing data
SELECT * FROM employees WHERE manager_id IS NULL;

-- ORDER BY: Sorts results
SELECT * FROM products ORDER BY price DESC;

-- LIMIT/OFFSET: Pagination
SELECT * FROM logs ORDER BY timestamp DESC LIMIT 10 OFFSET 10;
```

4. Subqueries & Nested Queries

```
-- Scalar Subquery: Returns a single value
SELECT name FROM employees
WHERE salary > (SELECT AVG(salary) FROM employees);

-- Subquery in FROM (used like a temporary table)
SELECT department, AVG(salary) FROM (
  SELECT * FROM employees WHERE status = 'active'
) AS active_employees
GROUP BY department;

-- EXISTS: Checks if subquery returns any rows
SELECT name FROM customers
WHERE EXISTS (
  SELECT 1 FROM orders WHERE orders.customer_id = customers.id
);
```

5. Useful Extras

SQL Syntax Cheat Sheet

```
-- GROUP BY: Aggregates data by groups
SELECT department, COUNT(*) FROM employees
GROUP BY department;

-- HAVING: Filters groups created by GROUP BY
SELECT department, COUNT(*) as total FROM employees
GROUP BY department
HAVING total > 5;

-- CASE: Implements conditional logic in queries
SELECT name, salary,
  CASE
    WHEN salary >= 100000 THEN 'High'
    WHEN salary >= 50000 THEN 'Medium'
    ELSE 'Low'
  END AS salary_level
FROM employees;

-- ALIASES: Gives temporary names for columns or tables
SELECT name AS employee_name FROM employees;
```