Keyword / Concept	Use Case / When to Use	Syntax / Example
SELECT	Select specific columns	SELECT col1, col2 FROM table;
SELECT *	Select all columns	SELECT * FROM table;
WHERE	Filter rows before aggregation	WHERE col = value
WHERE IS NULL	Filter NULL values	WHERE col IS NULL
WHERE IS NOT NULL	Filter NOT NULL values	WHERE col IS NOT NULL
LIKE	Pattern matching (substring)	WHERE col LIKE '%abc%'
ILIKE (Postgres)	Case-insensitive pattern matching	WHERE col ILIKE '%abc%'
REGEXP / ~	Regex pattern matching (MySQL/Postgres)	WHERE col REGEXP 'pattern' (MySQL) or WHERE col ~ 'pattern' (Postgres)
ORDER BY	Sort rows ascending/descending	ORDER BY col ASC
LIMIT	Limit number of rows returned	LIMIT n
OFFSET	Skip number of rows (pagination)	LIMIT n OFFSET m
INNER JOIN	Return rows matching in both tables	SELECT * FROM A INNER JOIN B ON A.id = B.a_id;
LEFT JOIN	All rows from left table + matched right	SELECT * FROM A LEFT JOIN B ON A.id = B.a_id;
RIGHT JOIN	All rows from right table + matched left	SELECT * FROM A RIGHT JOIN B ON A.id = B.a_id;
FULL OUTER JOIN	All rows from both tables, NULL if no match	SELECT * FROM A FULL OUTER JOIN B ON A.id = B.a_id;
CROSS JOIN	Cartesian product of two tables	SELECT * FROM A CROSS JOIN B;
SELF JOIN	Join table to itself	SELECT a.name, b.name FROM emp a JOIN emp b ON a.mgr_id = b.id;
COUNT()	Count rows or non-null values	SELECT COUNT(*) FROM table;
SUM()	Sum values of a numeric column	SELECT SUM(col) FROM table;
AVG()	Average value of a numeric column	SELECT AVG(col) FROM table;
MIN()	Minimum value	SELECT MIN(col) FROM table;
MAX()	Maximum value	SELECT MAX(col) FROM table;
GROUP BY	Group rows for aggregation	SELECT dept, SUM(salary) FROM employees GROUP BY dept;
HAVING	Filter groups after aggregation	HAVING SUM(salary) > 100000
DISTINCT	Return unique values	SELECT DISTINCT dept FROM employees;
SUM() OVER()	Running total / cumulative sum	SUM(col) OVER (PARTITION BY grp ORDER BY date ROWS UNBOUNDED PRECEDING)
AVG() OVER()	Moving average over window	AVG(col) OVER (ORDER BY date ROWS BETWEEN 2 PRECEDING AND CURRENT ROW)
ROW_NUMBER() OVER()	Unique row number per partition	ROW_NUMBER() OVER (PARTITION BY dept ORDER BY join_date)
RANK() OVER()	Rank with gaps	RANK() OVER (PARTITION BY dept ORDER BY salary DESC)
DENSE_RANK() OVER()	Rank without gaps	DENSE_RANK() OVER (ORDER BY salary DESC)
LAG()	Access previous row value	LAG(col, 1) OVER (ORDER BY date)
LEAD()	Access next row value	LEAD(col, 1) OVER (ORDER BY date)
NTH_VALUE()	Nth value in ordered window	NTH_VALUE(col, n) OVER (PARTITION BY dept ORDER BY date ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)
FIRST_VALUE()	First value in window	FIRST_VALUE(col) OVER (ORDER BY date)
LAST_VALUE()	Last value in window	LAST_VALUE(col) OVER (ORDER BY date)
RANGE	Logical frame for window function (value based)	SUM(col) OVER (ORDER BY date RANGE BETWEEN INTERVAL '7' DAY PRECEDING AND CURRENT ROW)
ROWS	Physical frame for window function (row based)	SUM(col) OVER (ORDER BY date ROWS BETWEEN 3 PRECEDING AND CURRENT ROW)
Scalar Subquery	Subquery returning single value	SELECT (SELECT MAX(salary) FROM employees) AS max_salary;
IN Subquery	Filter using values from subquery	WHERE dept_id IN (SELECT id FROM departments WHERE name='Sales')
EXISTS Subquery	Check if rows exist	WHERE EXISTS (SELECT 1 FROM orders WHERE orders.emp_id = employees.id)
CTE (WITH)	Named temporary result	WITH DeptSales AS (SELECT dept_id, SUM(sales) FROM orders GROUP BY dept_id) SELECT * FROM DeptSales WHERE SUM > 1000;
BEGIN TRANSACTION	Start transaction	BEGIN TRANSACTION;
COMMIT	Commit transaction	COMMIT;
ROLLBACK	Rollback transaction	ROLLBACK;
SAVEPOINT	Set savepoint in transaction	SAVEPOINT sp1;
ROLLBACK TO SAVEPOINT	Rollback to savepoint	ROLLBACK TO sp1;
PRIMARY KEY	Unique row identifier	id INT PRIMARY KEY
FOREIGN KEY	Reference to other table	FOREIGN KEY (dept_id) REFERENCES departments(id)
UNIQUE	Unique values in column	email VARCHAR(50) UNIQUE
NOT NULL	Column cannot be NULL	name VARCHAR(50) NOT NULL
DEFAULT	Default value if none provided	status VARCHAR(20) DEFAULT 'active'
CHECK	Constraint on values	CHECK (salary > 0)
LENGTH()	String length	LENGTH(col)
SUBSTRING()	Extract substring	SUBSTRING(col, start, length)
CONCAT()	Concatenate strings	CONCAT(col1, col2)
UPPER()	Uppercase string	UPPER(col)
LOWER()	Lowercase string	LOWER(col)
TRIM()	Trim whitespace	TRIM(col)
REPLACE()	Replace substring	REPLACE(col, 'old', 'new')
CURRENT_DATE / CURRENT_TIMES	Current date or datetime	CURRENT_DATE
EXTRACT()	Extract part from date/time	EXTRACT(YEAR FROM hire_date)
DATEDIFF()	Difference between dates	DATEDIFF(day, start_date, end_date)
DATEADD() / INTERVAL	Add time interval	hire_date + INTERVAL '7 days'
UNION	Combine distinct results	SELECT col FROM A UNION SELECT col FROM B
UNION ALL	Combine all results (duplicates kept)	SELECT col FROM A UNION ALL SELECT col FROM B
INTERSECT	Rows common in both queries	SELECT col FROM A INTERSECT SELECT col FROM B
EXCEPT / MINUS	Rows in first query but not second	SELECT col FROM A EXCEPT SELECT col FROM B
CASE	Conditional if-else logic	CASE WHEN condition THEN val1 ELSE val2 END
Conditional Aggregation	Sum or count with condition	SUM(CASE WHEN condition THEN 1 ELSE 0 END)
Alias (AS)	Rename columns or tables	SELECT col AS alias FROM table AS t
Comments	Add notes in query	-- single line or /* multi-line */
EXISTS	Check if subquery returns rows	WHERE EXISTS (SELECT 1 FROM table WHERE ...)