

SQL Cheat Sheet: Intermediate - LIKE, ORDER BY, GROUP BY, FUNCTIONS, Implicit JOIN



Command	Syntax	Description	Example
LIKE	<pre>SELECT column1, column2, ... FROM table_name WHERE columnN LIKE pattern;</pre>	<p>LIKE operator is used in a WHERE clause to search for a specified pattern in a column.</p> <p>There are two wildcards often used in conjunction with the LIKE operator which are percent sign(%) and underscore sign (_).</p>	<pre>SELECT f_name , l_name FROM employees WHERE address LIKE '%Elgin,IL%';</pre>
BETWEEN	<pre>SELECT column_name(s) FROM table_name WHERE column_name BETWEEN value1 AND value2;</pre>	<p>The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates. The BETWEEN operator is inclusive: begin and end values are included.</p>	<pre>SELECT * FROM employees WHERE salary BETWEEN 40000 AND 80000;</pre>
ORDER BY	<pre>SELECT column1, column2, ... FROM table_name ORDER BY column1, column2, ... ASC DESC;</pre>	<p>ORDER BY keyword is used to sort the result-set in ascending or descending order. The default is ascending.</p>	<pre>SELECT f_name, l_name, dep_id FROM employees ORDER BY dep_id DESC, l_name;</pre>
GROUP BY	<pre>SELECT column_name(s) FROM table_name WHERE condition GROUP BY column_name(s) ORDER BY column_name(s);</pre>	<p>GROUP BY clause is used in collaboration with the SELECT statement to arrange identical data into groups.</p>	<pre>SELECT dep_id, COUNT(*) FROM employees GROUP BY dep_id;</pre>
COUNT	<pre>SELECT COUNT(column_name) FROM table_name WHERE condition;</pre>	<p>COUNT function returns the number of rows that matches a specified criterion.</p>	<pre>SELECT COUNT(dep_id) FROM employees;</pre>
AVG	<pre>SELECT AVG(column_name) FROM table_name WHERE condition;</pre>	<p>AVG function returns the average value of a numeric column.</p>	<pre>SELECT AVG(salary) FROM employees;</pre>
SUM	<pre>SELECT SUM(column_name) FROM table_name WHERE condition;</pre>	<p>SUM function returns the total sum of a numeric column.</p>	<pre>SELECT SUM(salary) FROM employees;</pre>
MIN	<pre>SELECT MIN(column_name) FROM table_name WHERE condition;</pre>	<p>MIN function returns the smallest value of the SELECTed column.</p>	<pre>SELECT MIN(salary) FROM employees;</pre>
MAX	<pre>SELECT MAX(column_name) FROM</pre>	<p>MAX function returns the largest</p>	<pre>SELECT MAX(salary) FROM</pre>

	<code>table_name WHERE condition;</code>	value of the SELECTed column.	<code>employees;</code>
ROUND	<code>SELECT ROUND(2number, decimals, operation) AS RoundValue;</code>	ROUND function rounds a number to a specified number of decimal places.	<code>SELECT ROUND(salary) FROM employees;</code>
LENGTH	<code>SELECT LENGTH(column_name) FROM table;</code>	LENGTH function returns the length of a string (in bytes).	<code>SELECT LENGTH(f_name) FROM employees;</code>
UCASE	<code>SELECT UCASE(column_name) FROM table;</code>	UCASE function that displays the column name in each table in uppercase.	<code>SELECT UCASE(f_name) FROM employees;</code>
DISTINCT	<code>SELECT DISTINCT(column_name) FROM table;</code>	DISTINCT function is used to display data without duplicates.	<code>SELECT DISTINCT(UCASE(f_name)) FROM employees;</code>
DAY	<code>SELECT DAY(column_name) FROM table</code>	DAY function returns the day of the month for a given date	<code>SELECT DAY(b_date) FROM employees where emp_id = 'E1002';</code>
CURRENT DATE	<code>SELECT (CURRENT DATE - COLUMN) FROM table;</code>	CURRENT DATE is used to display the current date. This can be subtracted from the previous date to get the difference.	<code>SELECT YEAR(CURRENT DATE - b_date) As AGE, CURRENT_DATE, b_date FROM employees;</code>
Subquery	<code>SELECT column_name [, column_name] FROM table1 [, table2] WHERE column_name OPERATOR (SELECT column_name [, column_name] FROM table1 [, table2] [WHERE])</code>	Subquery is a query within another SQL query and embedded within the WHERE clause. A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.	<code>SELECT emp_id, fname, lname, salary FROM employees where salary < (SELECT AVG(salary) FROM employees);</code> <code>SELECT * FROM (SELECT emp_id, f_name, l_name, dep_id FROM employees) AS emp4all;</code> <code>SELECT * FROM employees WHERE job_id IN (SELECT job_id FROM jobs);</code>
Implicit Inner Join	<code>SELECT column_name(s) FROM table1, table2 WHERE table1.column_name = table2.column_name;</code>	Implicit Inner Join combines the two or more records but displays only matching values in both tables. Inner join applies only the specified columns.	<code>SELECT * FROM employees, jobs where employees.job_id = jobs.job_id;</code>
Implicit Cross Join	<code>SELECT column_name(s) FROM table1, table2;</code>	Implicit Cross Join defines as a Cartesian product where the number of rows in the first table multiplied by the number of rows in the second table..	<code>SELECT * FROM employees, jobs;</code>

Author(s)

[Lakshmi Holla](#)

Changelog

Date	Version	Changed by	Change Description
2021-07-28	1.0	Lakshmi Holla	Initial Version