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

Command	Syntax	Description	Example
LIKE	`SELECT column1, column2, ... FROM table_name WHERE columnN LIKE pattern;`	`LIKE` operator is used in a WHERE clause to search for a specified pattern in a column. There are two wildcards often used in conjunction with the LIKE operator which are percent sign(%) and underscore sign (_).	`SELECT f_name , l_name FROM employees WHERE address LIKE '%Elgin,IL%';`
BETWEEN	`SELECT column_name(s) FROM table_name WHERE column_name BETWEEN value1 AND value2;`	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.	`SELECT * FROM employees WHERE salary BETWEEN 40000 AND 80000;`
ORDER BY	`SELECT column1, column2, ... FROM table_name ORDER BY column1, column2, ... ASC|DESC;`	`ORDER BY` keyword is used to sort the result-set in ascending or descending order. The default is ascending.	`SELECT f_name, l_name, dep_id FROM employees ORDER BY dep_id DESC, l_name;`
GROUP BY	`SELECT column_name(s) FROM table_name WHERE condition GROUP BY column_name(s) ORDER BY column_name(s);`	`GROUP BY` clause is used in collaboration with the SELECT statement to arrange identical data into groups.	`SELECT dep_id, COUNT(*) FROM employees GROUP BY dep_id;`
COUNT	`SELECT COUNT(column_name) FROM table_name WHERE condition;`	`COUNT` function returns the number of rows that matches a specified criterion.	`SELECT COUNT(dep_id) FROM employees;`
AVG	`SELECT AVG(column_name) FROM table_name WHERE condition;`	`AVG` function returns the average value of a numeric column.	`SELECT AVG(salary) FROM employees;`
SUM	`SELECT SUM(column_name) FROM table_name WHERE condition;`	`SUM` function returns the total sum of a numeric column.	`SELECT SUM(salary) FROM employees;`
MIN	`SELECT MIN(column_name) FROM table_name WHERE condition;`	`MIN` function returns the smallest value of the SELECTed column.	`SELECT MIN(salary) FROM employees;`
MAX	`SELECT MAX(column_name) FROM table_name WHERE condition;`	`MAX` function returns the largest value of the SELECTed column.	`SELECT MAX(salary) FROM employees;`
ROUND	`SELECT ROUND(2number, decimals, operation) AS RoundValue;`	`ROUND` function rounds a number to a specified number of decimal places.	`SELECT ROUND(salary) FROM employees;`
LENGTH	`SELECT LENGTH(column_name) FROM table;`	`LENGTH` function returns the length of a string (in bytes).	`SELECT LENGTH(f_name) FROM employees;`

UCASE	`SELECT UCASE(column_name) FROM table;`	`UCASE` function that displays the column name in each table in uppercase.	`SELECT UCASE(f_name) FROM employees;`
DISTINCT	`SELECT DISTINCT(column_name) FROM table;`	`DISTINCT` function is used to display data without duplicates.	`SELECT DISTINCT(UCASE(f_name)) FROM employees;`
DAY	`SELECT DAY(column_name) FROM table`	`DAY` function returns the day of the month for a given date	`SELECT DAY(b_date) FROM employees where emp_id = 'E1002';`
CURRENT DATE	`SELECT (CURRENT DATE - COLUMN) FROM table;`	`CURRENT DATE` is used to display the current date.This can be subtracted from the previous date to get the difference.	`SELECT YEAR(CURRENT DATE - b_date) As AGE, CURRENT_DATE, b_date FROM employees;`

Subquery	`SELECT column_name [, column_name ] FROM table1 [, table2 ] WHERE column_name OPERATOR (SELECT column_name [, column_name ] FROM table1 [, table2 ] [WHERE])`	`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.
			`SELECT emp_id, fmame, lname, salary FROM employees where salary < (SELECT AVG(salary) FROM employees);`
			
			`SELECT * FROM ( SELECT emp_id, f_name, l_name, dep_id FROM employees) AS emp4all;`
			
			`SELECT * FROM employees WHERE job_id IN (SELECT job_ident FROM jobs);`

Implicit Inner Join	`SELECT column_name(s) FROM table1, table2 WHERE table1.column_name = table2.column_name;`	`Implicit Inner Join` combines the two or more records but displays only matching values in both tables. Inner join applies only the specified columns.	`SELECT * FROM employees, jobs where employees.job_id = jobs.job_ident;`
Implicit Cross Join	`SELECT column_name(s) FROM table1, table2;`	`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..	`SELECT * FROM employees, jobs;`

Author(s)

Lakshmi Holla

Changelog

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