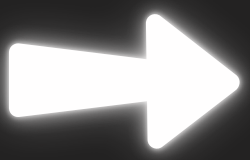
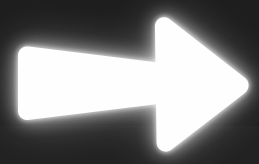
@Deepak Pal

All SQL Queries



ALL SQL Quaries

SQL Query	Query Description	Example
SELECT	Retrieves data from one or more tables.	`` `SELECT name, age FROM users;` ``
INSERT	Adds new records to a table.	`` `INSERT INTO users (name, age) VALUES ('Alice', 30);` ``
UPDATE	Modifies existing records in a table.	`` `UPDATE users SET age = 31 WHERE name = 'Alice';` ``
DELETE	Removes records from a table.	`` `DELETE FROM users WHERE name = 'Alice';` ``
CREATE TABLE	Creates a new table in the database.	`` `CREATE TABLE users (id INT PRIMARY KEY, name VARCHAR(100), age INT);` ``
ALTER TABLE	Modifies an existing table structure, such as adding or dropping columns.	`` `ALTER TABLE users ADD email VARCHAR(100);` ``
DROP TABLE	Deletes a table and all its data.	`` `DROP TABLE users;` ``
CREATE DATABASE	Creates a new database.	`` `CREATE DATABASE my_database;` ``
DROP DATABASE	Deletes a database and all its data.	`` `DROP DATABASE my_database;` ``
SELECT DISTINCT	Retrieves unique records from a table.	`` `SELECT DISTINCT age FROM users;` ``
WHERE	Filters records that meet a specific condition.	`` `SELECT name FROM users WHERE age > 25;` ``



WHERE	Filters records that meet a specific condition.	`SELECT name FROM users WHERE age > 25;`
ORDER BY	Sorts the result set by one or more columns.	`SELECT name, age FROM users ORDER BY age DESC;`
GROUP BY	Groups rows that have the same values into summary rows.	`SELECT age, COUNT(*) FROM users GROUP BY age;`
HAVING	Filters groups based on a condition.	`SELECT age, COUNT(*) FROM users GROUP BY age HAVING COUNT(*) > 1;`
JOIN	Combines rows from two or more tables based on a related column.	`SELECT users.name, orders.amount FROM users INNER JOIN orders ON users.id = orders.user_id;`
INNER JOIN	Retrieves records with matching values in both tables.	`SELECT users.name, orders.amount FROM users INNER JOIN orders ON users.id = orders.user_id;`
LEFT JOIN	Retrieves all records from the left table and matched records from the right table.	`SELECT users.name, orders.amount FROM users LEFT JOIN orders ON users.id = orders.user_id;`
RIGHT JOIN	Retrieves all records from the right table and matched records from the left table.	`SELECT users.name, orders.amount FROM users RIGHT JOIN orders ON users.id = orders.user_id;`
FULL OUTER JOIN	Retrieves records when there is a match in one of the tables.	`SELECT users.name, orders.amount FROM users FULL OUTER JOIN orders ON users.id = orders.user_id;`

UNION	Combines the result sets of two or more SELECT queries.	`SELECT name FROM users UNION SELECT name FROM employees;`
UNION ALL	Combines the result sets of two or more SELECT queries, including duplicates.	`SELECT name FROM users UNION ALL SELECT name FROM employees;`
SUBQUERY	A query nested inside another query.	`SELECT name FROM users WHERE age = (SELECT MAX(age) FROM users);`
EXISTS	Checks if a subquery returns any records.	`SELECT name FROM users WHERE EXISTS (SELECT 1 FROM orders WHERE users.id = orders.user_id);`
ANY	Compares a value to any value in a list or subquery.	`SELECT name FROM users WHERE age > ANY (SELECT age FROM users WHERE age < 30);`
ALL	Compares a value to all values in a list or subquery.	`SELECT name FROM users WHERE age > ALL (SELECT age FROM users WHERE age < 30);`
LIKE	Searches for a specified pattern in a column.	`SELECT name FROM users WHERE name LIKE 'A%';`
IN	Checks if a value is within a set of values.	`SELECT name FROM users WHERE age IN (25, 30, 35);`
BETWEEN	Selects values within a range.	`SELECT name FROM users WHERE age BETWEEN 20 AND 30;`
LIMIT	Specifies the number of records to return.	`SELECT name FROM users LIMIT 5;`

LIMIT	Specifies the number of records to return.	`SELECT name FROM users LIMIT 5;`
OFFSET	Specifies the starting point for records to return.	`SELECT name FROM users LIMIT 5 OFFSET 10;`
TRUNCATE TABLE	Removes all records from a table but retains the table structure.	`TRUNCATE TABLE users;`
CREATE INDEX	Creates an index on a table to improve query performance.	`CREATE INDEX idx_name ON users (name);`
DROP INDEX	Deletes an index from a table.	`DROP INDEX idx_name;`
CASE	Provides conditional logic in SQL queries.	`SELECT name, age, CASE WHEN age < 30 THEN 'Young' ELSE 'Old' END as age_group FROM users;`
CAST	Converts a value from one data type to another.	`SELECT name, CAST(age AS CHAR) FROM users;`

Was it helpful?

follow me for more!

Like Comment Share Save