Vikram
@code._learning

100 SQL Commands

1

- **SELECT** - retrieves data from a database
- **INSERT** - inserts new data into a database
- **UPDATE** - updates existing data in a database
- **DELETE** - deletes data from a database
- **CREATE DATABASE** - creates a new database
- **CREATE TABLE** - creates a new table in a database
- **ALTER TABLE** - modifies an existing table structure

- **DROP TABLE** - deletes a table from a database
- **TRUNCATE TABLE** - removes all records from a table
- **CREATE INDEX** - creates an index on a table
- **DROP INDEX** - deletes an index from a table
- **JOIN** - combines rows from two or more tables based on a related column
- **INNER JOIN** - returns rows when there is a match in both tables
- **LEFT JOIN** - returns all rows from the left table, and the matched rows from the right table
- **RIGHT JOIN** - returns all rows from the right table, and the matched rows from the left table

- **FULL JOIN** - returns rows when there is a match in one of the tables
- **UNION** - combines the results of two or more SELECT statements
- **UNION ALL** - combines the results of two or more SELECT statements, including duplicates
- **GROUP BY** - groups rows that have the same values into summary rows
- **HAVING** - filters records based on a specified condition
- **ORDER BY** - sorts the result set in ascending or descending order
- **COUNT** - returns the number of rows that satisfy the condition
- **SUM** - calculates the sum of a set of values
- **AVG** - calculates the average of a set of values
- **MIN** - returns the smallest value in a set of values
- **MAX** - returns the largest value in a set of values
- **DISTINCT** - selects unique values from a column
- **WHERE** - filters records based on specified conditions

- **AND** - combines multiple conditions in a WHERE clause
- **OR** - specifies multiple alternative conditions in a WHERE clause
- **NOT** - negates a condition in a WHERE clause
- **BETWEEN** - selects values within a specified range
- **IN** - specifies multiple values for a column
- **LIKE** - selects rows that match a specified pattern
- **IS NULL** - checks for NULL values in a column
- **IS NOT NULL** - checks for non-NULL values in a column

- **EXISTS** - tests for the existence of any record in a subquery
- **CASE** - performs conditional logic in SQL statements
- **WHEN** - specifies conditions in a CASE statement
- **THEN** - specifies the result if a condition is true in a CASE statement
- **ELSE** - specifies the result if no condition is true in a CASE statement
- **END** - ends the CASE statement

- **PRIMARY KEY** - uniquely identifies each record in a table
- **FOREIGN KEY** - establishes a relationship between tables
- **CONSTRAINT** - enforces rules for data in a table
- **DEFAULT** - specifies a default value for a column
- **NOT NULL** - ensures that a column cannot contain NULL values
- **UNIQUE** - ensures that all values in a column are unique
- **CHECK** - enforces a condition on the values in a column

- **CASCADE** - automatically performs a specified action on related records
- **SET NULL** - sets the value of foreign key columns to NULL when a referenced record is deleted
- **SET DEFAULT** - sets the value of foreign key columns to their default value when a referenced record is deleted
- **NO ACTION** - specifies that no action should be taken on related records when a referenced record is deleted

- **RESTRICT** - restricts the deletion of a referenced record if there are related records
- **CASE WHEN** - conditional expression in SELECT statements
- **WITH** - defines a common table expression (CTE)
- **INTO** - specifies a target table for the result set of a SELECT statement
- **TOP** - limits the number of rows returned by a query
- **LIMIT** - limits the number of rows returned by a query (used in some SQL dialects)
- **OFFSET** - specifies the number of rows to skip before starting to return rows
- **FETCH** - retrieves rows from a result set one at a time
- **ROW_NUMBER()** - assigns a unique sequential integer to each row in a result set
- **RANK()** - assigns a unique rank to each row in a result set, with gaps in the ranking sequence possible
- **DENSE_RANK()** - assigns a unique rank to each row in a result set, with no gaps in the ranking sequence

- **NTILE()** - divides the result set into a specified number of equally sized groups
- **LEAD()** - retrieves the value from the next row in a result set
- **LAG()** - retrieves the value from the previous row in a result set
- **PARTITION BY** - divides the result set into partitions to which the window function is applied separately
- **ORDER BY** - specifies the order of rows within each partition for window functions
- **ROWS** - specifies the window frame for window functions

- **RANGE** - specifies the window frame based on values rather than rows for window functions
- **CURRENT_TIMESTAMP** - returns the current date and time
- **CURRENT_DATE** - returns the current date
- **CURRENT_TIME** - returns the current time
- **DATEADD** - adds a specified time interval to a date
- **DATEDIFF** - calculates the difference between two dates

7

- **DATEPART** - extracts a specific part of a date
- **GETDATE** - returns the current date and time (similar to CURRENT_TIMESTAMP)
- **GROUPING SETS** - specifies multiple groupings for aggregation
- **CUBE** - generates all possible combinations of grouping sets for aggregation
- **ROLLUP** - generates subtotal values for a hierarchy of values
- **INTERSECT** - returns the intersection of two result sets
- **EXCEPT** - returns the difference between two result sets
- **MERGE** - performs insert, update, or delete operations on a target table based on the results of a join with a source table
- **CROSS APPLY** - performs a correlated subquery against each row of the outer table
- **OUTER APPLY** - similar to CROSS APPLY, but also returns rows from the outer table that have no matching rows in the inner table
- **PIVOT** - rotates a table-valued expression by turning the unique values from one column into multiple columns in the output

- **UNPIVOT** - rotates a table-valued expression by turning multiple columns into unique rows in the output
- **COALESCE** - returns the first non-NULL expression in a list
- **NULLIF** - returns NULL if the two specified expressions are equal, otherwise returns the first expression
- **IIF** - returns one of two values based on a Boolean expression

- **CONCAT** - concatenates two or more strings
- **SUBSTRING** - extracts a substring from a string
- **CHARINDEX** - finds the position of a substring within a string
- **REPLACE** - replaces all occurrences of a specified substring within a string with another substring
- **LEN** - returns the length of a string
- **UPPER** - converts a string to uppercase
- **LOWER** - converts a string to lowercase
- **TRIM** - removes leading and trailing spaces from a string
- **ROUND** - rounds a numeric value to a specified number of decimal places