100 SQL Commands

1. **SELECT**
    - retrieves data from a database
2. **INSERT**
    - inserts new data into a database
3. **UPDATE**
    - updates existing data in a database
4. **DELETE**
    - deletes data from a database
5. **CREATE DATABASE**
    - creates a new database
6. **CREATE TABLE**
    - creates a new table in a database
7. **ALTER TABLE**
    - modifies an existing table structure
8. **DROP TABLE**
    - deletes a table from a database
9. **TRUNCATE TABLE**
    - removes all records from a table
10. **CREATE INDEX**
    - creates an index on a table
11. **DROP INDEX**
    - deletes an index from a table

12. **JOIN**

    - combines rows from two or more tables based on a related column

13. **INNER JOIN**

    - returns rows when there is a match in both tables

14. **LEFT JOIN**

    - returns all rows from the left table, and the matched rows from the right table

15. **RIGHT JOIN**

    - returns all rows from the right table, and the matched rows from the left table

16. **FULL JOIN**

    - returns rows when there is a match in one of the tables

17. **UNION**

    - combines the results of two or more SELECT statements

18. **UNION ALL**

    - combines the results of two or more SELECT statements, including duplicates

19. **GROUP BY**

    - groups rows that have the same values into summary rows @code._learning

20. **HAVING**

    - filters records based on a specified condition

21. **ORDER BY**

    - sorts the result set in ascending or descending order

22. **COUNT**

    - returns the number of rows that satisfy the condition

23. **SUM**

    - calculates the sum of a set of values

24. **AVG**
    - calculates the average of a set of values

25. **MIN**
    - returns the smallest value in a set of values

26. **MAX**
    - returns the largest value in a set of values

27. **DISTINCT**
    - selects unique values from a column

28. **WHERE**
    - filters records based on specified conditions

29. **AND**
    - combines multiple conditions in a WHERE clause

30. **OR**
    - specifies multiple alternative conditions in a WHERE clause

31. **NOT**
    - negates a condition in a WHERE clause

32. **BETWEEN**
    - selects values within a specified range

33. **IN**
    - specifies multiple values for a column

34. **LIKE**
    - selects rows that match a specified pattern

35. **IS NULL**
    - checks for NULL values in a column
36. **IS NOT NULL**
    - checks for non-NULL values in a column
37. **EXISTS**
    - tests for the existence of any record in a subquery
38. **CASE**
    - performs conditional logic in SQL statements
39. **WHEN**
    - specifies conditions in a CASE statement
40. **THEN**
    - specifies the result if a condition is true in a CASE statement
41. **ELSE**
    - specifies the result if no condition is true in a CASE statement
42. **END**
    - ends the CASE statement
43. **PRIMARY KEY**
    - uniquely identifies each record in a table
44. **FOREIGN KEY**
    - establishes a relationship between tables
45. **CONSTRAINT**
    - enforces rules for data in a table
46. **DEFAULT**

- specifies a default value for a column

47. **NOT NULL**

    - ensures that a column cannot contain NULL values

48. **UNIQUE**

    - ensures that all values in a column are unique

49. **CHECK**

    - enforces a condition on the values in a column

50. **CASCADE**

    - automatically performs a specified action on related records

51. **SET NULL**

    - sets the value of foreign key columns to NULL when a referenced record is deleted

52. **SET DEFAULT**

    - sets the value of foreign key columns to their default value when a referenced record is deleted

53. **NO ACTION**

    - specifies that no action should be taken on related records when a referenced record is deleted

54. **RESTRICT**

    - restricts the deletion of a referenced record if there are related records

55. **CASE WHEN**

    - conditional expression in SELECT statements

56. **WITH**

    - defines a common table expression (CTE)

57. **INTO**

    - specifies a target table for the result set of a SELECT statement

58. **TOP**
    - limits the number of rows returned by a query
59. **LIMIT**
    - limits the number of rows returned by a query (used in some SQL dialects)
60. **OFFSET**
    - specifies the number of rows to skip before starting to return rows
61. **FETCH**
    - retrieves rows from a result set one at a time
62. **ROW_NUMBER()**
    - assigns a unique sequential integer to each row in a result set
63. **RANK()**
    - assigns a unique rank to each row in a result set, with gaps in the ranking sequence possible
64. **DENSE_RANK()**
    - assigns a unique rank to each row in a result set, with no gaps in the ranking sequence
65. **NTILE()**
    - divides the result set into a specified number of equally sized groups
66. **LEAD()**
    - retrieves the value from the next row in a result set
67. **LAG()**
    - retrieves the value from the previous row in a result set
68. **PARTITION BY**
    - divides the result set into partitions to which the window function is applied separately
69. **ORDER BY**

- specifies the order of rows within each partition for window functions

70. **ROWS**
    - specifies the window frame for window functions
71. **RANGE**
    - specifies the window frame based on values rather than rows for window functions
72. **CURRENT TIMESTAMP**
    - returns the current date and time
73. **CURRENT_DATE**
    - returns the current date
74. **CURRENT_TIME**
    - returns the current time
75. **DATEADD**
    - adds a specified time interval to a date
76. **DATEDIFF**
    - calculates the difference between two dates
77. **DATEPART**
    - extracts a specific part of a date
78. **GETDATE**
    - returns the current date and time (similar to CURRENT_TIMESTAMP)
79. **GROUPING SETS**
    - specifies multiple groupings for aggregation
80. **CUBE**
    - generates all possible combinations of grouping sets for aggregation

81. **ROLLUP**
    - generates subtotal values for a hierarchy of values
82. **INTERSECT**
    - returns the intersection of two result sets
83. **EXCEPT**
    - returns the difference between two result sets
84. **MERGE**
    - performs insert, update, or delete operations on a target table based on the results of a join with a source table
85. **CROSS APPLY**
    - performs a correlated subquery against each row of the outer table
86. **OUTER APPLY**
    - similar to CROSS APPLY, but also returns rows from the outer table that have no matching rows in the inner table
87. **PIVOT**
    - rotates a table-valued expression by turning the unique values from one column into multiple columns in the output
88. **UNPIVOT**
    - rotates a table-valued expression by turning multiple columns into unique rows in the output
89. **COALESCE**
    - returns the first non-NULL expression in a list
90. **NULLIF**
    - returns NULL if the two specified expressions are equal, otherwise returns the first expression
91. **IIF**
    - returns one of two values based on a Boolean expression

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