1. Basic SQL Syntax:

* SQL keywords
* Data types
* Operators
* SQL statements (SELECT, INSERT, UPDATE, DELETE)
* <https://lnkd.in/dB2nWGCC>
* <https://lnkd.in/g26GBTPs>

1. Data Definition Language (DDL):

* CREATE TABLE
* ALTER TABLE
* DROP TABLE
* Truncate table
* <https://lnkd.in/dmjMAvxD>
* <https://lnkd.in/dE7zygpJ>

1. Data Manipulation Language (DML):

* SELECT statement (SELECT, FROM, WHERE, ORDER BY, GROUP BY, HAVING, JOINs)
* INSERT statement
* UPDATE statement
* DELETE statement
* <https://lnkd.in/dwZtQrDm>
* <https://sqlzoo.net/>

1. Aggregate Functions:

* SUM, AVG, COUNT, MIN, MAX
* GROUP BY clause
* HAVING clause
* <https://lnkd.in/dcePRpcT>
* <https://learnsql.com/>

1. Data Constraints:

* Primary Key
* Foreign Key
* Unique
* NOT NULL
* CHECK
* <https://lnkd.in/d_dNcGGT>

1. Joins:

* INNER JOIN
* LEFT JOIN
* RIGHT JOIN
* FULL OUTER JOIN
* Self Join
* Cross Join
* <https://lnkd.in/dNaZb69H>

7. Subqueries:  
Types of subqueries (scalar, column, row, table)  
Nested subqueries  
Correlated subqueries  
  
<https://lnkd.in/dy5cNkrG>  
  
8. Advanced SQL Functions:  
String functions (CONCAT, LENGTH, SUBSTRING, REPLACE, UPPER, LOWER)  
Date and time functions (DATE, TIME, TIMESTAMP, DATEPART, DATEADD)  
Numeric functions (ROUND, CEILING, FLOOR, ABS, MOD)  
Conditional functions (CASE, COALESCE, NULLIF)  
  
<https://lnkd.in/d3F-MTwu>  
  
<https://lnkd.in/dxj7ir9Z>  
  
9. Views:  
Creating views  
Modifying views  
Dropping views  
  
<https://lnkd.in/d7szhPJT>  
  
10. Indexes:  
Creating indexes  
Using indexes for query optimization  
  
<https://lnkd.in/d-pETDUG>  
  
11. Transactions:  
ACID properties  
Transaction management (BEGIN, COMMIT, ROLLBACK, SAVEPOINT)  
Transaction isolation levels  
  
<https://lnkd.in/dqXffY6N>  
  
12. Data Integrity and Security:  
Data integrity constraints (referential integrity, entity integrity)  
GRANT and REVOKE statements (granting and revoking permissions)  
Database security best practices  
  
<https://lnkd.in/d_dNcGGT>  
  
13. Stored Procedures and Functions:  
Creating stored procedures  
Executing stored procedures  
Creating functions  
Using functions in queries  
  
<https://lnkd.in/d-su5yT8>  
  
14. Performance Optimization:  
Query optimization techniques (using indexes, optimizing joins, reducing subqueries)  
Performance tuning best practices  
  
<https://lnkd.in/dZERrBBX>  
  
15. Advanced SQL Concepts:  
  
Recursive queries  
Pivot and unpivot operations  
Window functions (Row\_number, rank, dense\_rank, lead & lag)  
CTEs (Common Table Expressions)  
Dynamic SQL  
  
<https://lnkd.in/d8ykNRfG>  
  
<https://lnkd.in/dBWrf4Mg>

# List of SQL Keywords

1. **CREATE:** The CREATE Keyword is used to create a database, table, views, and index. We can create the table CUSTOMER as below.

create table CUSTOMER (

id INT PRIMARY KEY,

name VARCHAR(50),

state VARCHAR(20) );

1. **PRIMARY KEY:** This keyword uniquely identifies each of the records.

create DATABASE User;

1. **INSERT:** The INSERT Keyword is used to insert the rows of data into a table. We can insert the rows below to the already created CUSTOMER table using the queries below.

insert into CUSTOMER values (121,'Rajesh','Maharashtra');

insert into CUSTOMER values (256,'Leela','Punjab');

1. **SELECT:** This keyword is used to select the data from the database or table. The ‘\*’ is used in the select statement to select all the columns in a table.

select NAME from CUSTOMER;

**5. FROM**

The keyword indicates the table from which the data is selected or deleted.

**6. ALTER**

The Keyword ALTER is used to modify the columns in tables. The ALTER COLUMN statement modifies the data type of a column, and the ALTER TABLE modifies the columns by adding or deleting them.

We can modify the columns of the CUSTOMER table as below by adding a new column, “AGE”.

ALTER TABLE CUSTOMER ADD AGE INT;

SELECT \* FROM CUSTOMER;

This query above will add the new column “AGE” with values for all the rows as null. Also, the above statement uses another SQL keyword ‘ADD’.

**7. ADD**

This is used to add a column to the existing table.

**8. DISTINCT**

The keyword DISTINCT is used to select distinct values. We can use SELECT DISTINCT to select only the distinct values from a table.

Let us add a duplicate value for the state Punjab as below:

INSERT INTO CUSTOMER VALUES(178, 'Pooja', 'Punjab','null');

The customer table now has the below rows:

Now we can see the distinct values for the column STATE by using the below query:

SELECT DISTINCT(STATE) FROM CUSTOMER;

**9. UPDATE**

This keyword is used in an SQL statement to update the existing rows in a table.

UPDATE CUSTOMER SET STATE ='Rajasthan' WHERE CUST\_ID= 121;

SELECT \* FROM CUSTOMER;

The CUST\_ID with value 121 is updated with a new state Rajasthan.

**10. SET**

This Keyword is used to specify the column or values to be updated.

**11. DELETE**

This is used to delete the existing rows from a table.

DELETE FROM CUSTOMER WHERE NAME='Rajesh';

The above query will display the below as the row with Name as Rajesh is deleted from the result set.

While using the DELETE keyword, if we do not use the WHERE clause, all the records will be deleted from the table.

DELETE FROM CUSTOMER;

The above query will delete all the records of the CUSTOMER table.

**12. TRUNCATE**

This is used to delete the data in a table, but it does not delete the structure of the table.

TRUNCATE TABLE CUSTOMER;

The above query only deletes the data, but the structure of the table remains. So there is no need to re-create the table.

**13. AS**

The Keyword AS is used as an alias to rename the column or table.

SELECT CUST\_ID AS CUSTOMER\_ID, NAME AS CUSTOMER\_NAME FROM CUSTOMER;

The above statement will create the alias for the columns CUST\_ID and NAME as below:

**14. ORDER BY**

This is used to sort the result in descending or ascending order. This sorts the result by default in ascending order.

**15. ASC**

This keyword is used for sorting the data returned by the SQL query in ascending order.

SELECT \* FROM CUSTOMER ORDER BY NAME ASC;

The above query will select all the columns from the CUSTOMER table and sorts the data by the NAME column in ascending order.

**16. DESC**

This keyword is to sort the result set in descending order.

SELECT \* FROM CUSTOMER ORDER BY CUST\_ID DESC;

The above query will sort all the selected fields of the table in the descending order of CUST\_ID.

**17. BETWEEN**

This keyword is used to select values within a given range. The below query uses the BETWEEN keyword to select the CUST\_ID and NAME within a given range of values for the CUST\_ID.

SELECT CUST\_ID, NAME FROM CUSTOMER WHERE CUST\_ID BETWEEN 100 AND 500;

The above query will give the below result

**18. WHERE**

This keyword is used to filter the result set so that only the values satisfying the condition are included.

SELECT \* FROM CUSTOMER WHERE STATE ='Punjab';

The above query selects all the values from the table for which the state is Punjab.

**19. AND**

This keyword is used along with the WHERE clause to select the rows for which both conditions are true.

SELECT \* FROM CUSTOMER WHERE STATE ='Punjab' AND CUST\_ID= 256;

The above query will give the result as mentioned below.

But if one of the conditions is not satisfied, then the query will not return any result, as stated in the below query.

SELECT \* FROM CUSTOMER WHERE STATE ='Punjab' AND CUST\_ID= 121;

**20. OR**

This is used with the WHERE clause to include the rows in the result set in case of either condition is true.

The below SQL statement will select the fields from the CUSTOMER table if the state is Punjab or UP.

SELECT \* FROM CUSTOMER WHERE STATE='Punjab' OR STATE='UP';

In the case of the OR keyword, we can see from the above result that if any of the given conditions are true, that gets included in the result set.

**21. NOT**

The keyword NOT uses a WHERE clause to include the rows in the result set where a condition is not true.

We can use the NOT keyword in the below query to not include the rows from the state Punjab as below.

SELECT \* FROM CUSTOMER WHERE NOT STATE = 'Punjab';

The query will return the rows with the other states, excluding Punjab in the result set as below:

**22. LIMIT**

This keyword retrieves the records from the table to limit them based on the limit value.

SELECT \* FROM CUSTOMER LIMIT 3;

The above query will select the records from the table CUSTOMER, but it will display only the 3 rows of data from the table as below

**23. IS NULL**

The keyword IS NULL is used to check for NULL values.

The below query will show all the records for which the AGE column has NULL values.

SELECT \* FROM CUSTOMER WHERE AGE IS NULL;

IS NOT NULL

This is used to search the NOT NULL values.

SELECT \* FROM CUSTOMER WHERE STATE IS NOT NULL;

As the column STATE has no null values, the above query will show the below result.

**24. DROP**

The DROP keyword can be used to delete a database, table, view, column, index, etc.

**25. DROP COLUMN**

We can delete an existing column in a table using a DROP COLUMN and an ALTER statement. Let us delete the column AGE by using the below query.

ALTER TABLE CUSTOMER DROP COLUMN AGE;

We can see that in the above result, the AGE column is dropped.

**26. DROP DATABASE**

A database in SQL can be deleted by using the DROP DATABASE statement.

DROP DATABASE DATABASE\_NAME;

**27. DROP TABLE**

A table in SQL can be deleted by using a DROP TABLE statement.

DROP TABLE TABLE\_NAME;

We can delete the table CUSTOMER by using the DROP TABLE keyword as below.

But we must be careful while using the DROP TABLE as it will remove the table definition, all the data, indexes, etc.

**28. GROUP BY**

This is used along with the aggregate functions like COUNT, MAX, MIN, AVG, SUM, etc., and groups the result set. The below query will group the CUST\_ID according to the various states.

SELECT COUNT(CUST\_ID),STATE FROM CUSTOMER GROUP BY STATE;

The result shows the count of different CUST\_ID grouped by states.

**29. HAVING**

This keyword is used with aggregate functions and GROUP BY instead of the WHERE clause to filter the values of a result set.

SELECT COUNT(CUST\_ID),STATE FROM CUSTOMER GROUP BY STATE HAVING COUNT(CUST\_ID)>=2;

The above query will filter the result set by displaying only those values which satisfy the condition given in the [HAVING clause](https://www.educba.com/oracle-having-clause/).

The above result set shows the values for which the count of the customer ids is more than 2.

**30. IN**

The IN keyword is used within a WHERE clause to specify more than 1 value, or we can say that it can be used instead of the usage of multiple OR keyword in a query.

The query below will select the records for the states Maharashtra, Punjab, and UP by using the IN keyword.

SELECT \* FROM CUSTOMER WHERE STATE IN ('Maharashtra','Punjab','UP');

The above result shows the usage of IN keyword, which selects the records only for the states specified within the IN clause.

**31. JOIN**

The keyword JOIN combines the rows between two or more tables with related columns among the tables. The JOIN can be INNER, LEFT, RIGHT, OUTER JOIN, etc.

Lets us take another table, ‘CUST\_ORDER’, as an example.

We can perform an inner join of the CUSTOMER and CUST\_ORDER tables as below.

SELECT CUSTOMER.NAME, CUSTOMER.STATE, CUST\_ORDER.ITEM\_DES

FROM CUSTOMER INNER JOIN CUST\_ORDER

ON CUSTOMER.CUST\_ID =CUST\_ORDER.ID;

The above query will join the two tables CUSTOMER and CUST\_ORDER on the columns CUST\_ID and ID and display only the values which are present in both tables.

This result shows the matching records for cust\_id 121,908 and 178, which are common in both tables. But the other cust\_ids are excluded as they are not present in the CUST\_ORDER table. Similarly, the other JOINs can be performed.

**32. UNION**

The UNION keyword combines the distinct values of two or more select statements.

SELECT CUST\_ID FROM CUSTOMER UNION SELECT ID FROM CUST\_ORDER;

The above query will show the below result.

**33. UNION ALL**

This keyword combines two or more select statements but allows duplicate values.

SELECT CUST\_ID FROM CUSTOMER UNION ALL SELECT ID FROM CUST\_ORDER;

The above result shows that UNION ALL allows duplicate values which would not be present in the case of UNION.

**34. EXISTS**

The keyword EXISTS checks if a certain record exists in a sub-query.

SELECT NAME FROM CUSTOMER WHERE EXISTS (SELECT ITEM\_DES FROM CUST\_ORDER WHERE CUST\_ID = ID);

The above query will return true as the sub-query returns the below values.

**35. LIKE**

This keyword is used to search along with a WHERE clause for a particular pattern. Wildcard % is used to search for a pattern.

In the below query, let us search for a pattern ‘ya’ which occurs in the column ‘NAME’.

SELECT NAME FROM CUSTOMER WHERE NAME LIKE '%ya';

**36. CASE**

This keyword is used to display different outputs according to different conditions.

SELECT CUST\_ID, NAME,

CASE WHEN STATE = 'Punjab' THEN "State is Punjab"

ELSE "State is NOT Punjab"

END AS Output

FROM CUSTOMER;