SQL Out Come

1. SELECT Statement:

Retrieving data from a table. The SELECT statement is used to retrieve data from a table in SQL. It allows you to specify the columns to be selected and the table from which to retrieve data. Example:

```
SELECT column1, column2 FROM table_name;
```

2. DELETE Statement:

Removing data from a table. The DELETE statement is used to remove data from a table in SQL. It allows you to specify the table from which to delete data and optional conditions to filter the rows to be deleted. Example:

```
DELETE FROM table_name WHERE condition;
```

3. WHERE Clause:

Filtering data based on conditions. The WHERE clause is used in SQL statements to filter data based on specified conditions. It allows you to retrieve only the rows that satisfy the given condition. Example:

```
SELECT column1, column2 FROM table_name WHERE condition;
```

4. ORDER BY Clause:

Sorting data in ascending or descending order.The ORDER BY clause is used to sort the result set in SQL. It allows you to specify one or more columns and the sorting order (ASC for ascending, DESC for descending). Example:

```
SELECT column1, column2 FROM table_name ORDER BY column1 ASC;
```

5. GROUP BY Clause:

Grouping data based on one or more columns. The GROUP BY clause is used to group rows in SQL based on specified columns. It is often used with aggregate functions to perform calculations on grouped data. Example:

```
SELECT column1, COUNT(*) FROM table_name GROUP BY column1;
```

6. HAVING Clause:

Filtering data after grouping. The HAVING clause is used in SQL statements to filter data after the GROUP BY clause has been applied. It allows you to specify conditions on aggregated values. Example:

SELECT column1, COUNT(*) FROM table_name GROUP BY column1 HAVING COUNT(*) > 10;

7. JOIN:

Combining data from multiple tables. JOIN is used to combine data from multiple tables based on related columns. It allows you to retrieve data from multiple tables in a single query. Example:

SELECT table1.column1, table2.column2 FROM table1 JOIN table2 ON table1.column_key = table2.column_key;

8. INNER JOIN:

Retrieving records with matching values in both tables. INNER JOIN returns only the matching records between two tables based on a specified condition. It selects records that have matching values in both tables. Example:

SELECT column1, column2 FROM table1 INNER JOIN table2 ON table1.column_key = table2.column_key;

9. LEFT JOIN:

Retrieving all records from the left table and matching records from the right table. LEFT JOIN returns all records from the left table and the matching records from the right table based on a specified condition. Example:

SELECT column1, column2 FROM table1 LEFT JOIN table2 ON table1.column_key = table2.column_key;

10. RIGHT JOIN:

Retrieving all records from the right table and matching records from the left table. RIGHT JOIN returns all records from the right table and the matching records from the left table based on a specified condition. Example:

SELECT column1, column2 FROM table1 RIGHT JOIN table2 ON table1.column_key = table2.column_key;

11. FULL OUTER JOIN:

Retrieving all records from both tables. FULL OUTER JOIN returns all records from both tables, including unmatched rows. It combines the results of LEFT JOIN and RIGHT JOIN. Example:

SELECT column1, column2 FROM table1 FULL OUTER JOIN table2 ON table1.column_key = table2.column_key;

12. UNION:

Combining the result sets of two or more SELECT statements. UNION is used to combine the results of two or more SELECT statements into a single result set. It removes duplicate rows by default. Example:

```
SELECT column1, column2 FROM table1 UNION SELECT column1, column2 FROM table2;
```

13. DISTINCT Keyword:

Retrieving unique values from a column. The DISTINCT keyword is used to retrieve unique values from a column in SQL. It eliminates duplicate values from the result set. Example:

```
SELECT DISTINCT column1 FROM table_name;
```

14. NULL Values:

Handling records with missing or unknown values. NULL is a special value in SQL that represents missing or unknown data. It is used to indicate the absence of a value in a column. Example:

```
SELECT column1, column2 FROM table_name WHERE column1 IS NULL;
```

15. LIKE Operator:

Matching patterns in data. The LIKE operator is used to match patterns in data using wildcard characters (% and _). It is often used with the WHERE clause. Example:

```
SELECT column1, column2 FROM table_name WHERE column1 LIKE 'A%';
```

16. BETWEEN Operator:

Selecting values within a specific range. The BETWEEN operator is used to select values within a specified range, inclusive of the endpoints. It is often used with the WHERE clause. Example:

```
SELECT column1, column2 FROM table_name WHERE column1 BETWEEN 10 AND 20;
```

17. IN Operator:

Matching values against a list of multiple values. The IN operator is used to match values against a list of multiple values in SQL. It is often used with the WHERE clause. Example:

```
SELECT column1, column2 FROM table_name WHERE column1 IN (value1, value2, value3);
```

18. Functions:

Using built-in functions for data manipulation and calculations. SQL provides various built-in functions for data manipulation and calculations. Examples include COUNT, SUM, AVG, MAX, MIN. Example:

```
SELECT COUNT(column1) FROM table_name;
```

19. Subqueries:

Using queries within queries. Subqueries are queries nested within another query. They are used to retrieve data based on the results of another query. Example:

```
SELECT column1, column2 FROM table_name WHERE column1 IN (SELECT column1
FROM table2);
```

20. Aliases:

Assigning temporary names to tables or columns. Aliases are used to assign temporary names to tables or columns in SQL. They make queries more readable and concise. Example:

```
SELECT column1 AS alias1, column2 AS alias2 FROM table_name AS alias;
```