SQL (Structured Query Language) is used for managing and manipulating relational databases. The syntax of SQL can be quite comprehensive, covering various operations like querying, updating, inserting, and deleting data. Below is an overview of common SQL commands along with examples to illustrate their usage.

**1. SELECT Statement**

Used to retrieve data from one or more tables.

SELECT column1, column2, ...

FROM table\_name

WHERE condition

ORDER BY column

LIMIT number;

**Example:**

SELECT first\_name, last\_name

FROM employees

WHERE department = 'Sales'

ORDER BY last\_name

LIMIT 10;

**2. INSERT INTO Statement**

Used to insert new records into a table.

INSERT INTO table\_name (column1, column2, ...)

VALUES (value1, value2, ...);

**Example:**

INSERT INTO employees (first\_name, last\_name, department, hire\_date)

VALUES ('John', 'Doe', 'Sales', '2024-08-27');

**3. UPDATE Statement**

Used to modify existing records.

UPDATE table\_name

SET column1 = value1, column2 = value2, ...

WHERE condition;

**Example:**

UPDATE employees

SET department = 'Marketing'

WHERE last\_name = 'Doe';

**4. DELETE Statement**

Used to delete records from a table.

DELETE FROM table\_name

WHERE condition;

**Example:**

DELETE FROM employees

WHERE last\_name = 'Doe';

**5. CREATE TABLE Statement**

Used to create a new table in the database.

CREATE TABLE table\_name (

column1 datatype constraints,

column2 datatype constraints,

...

);

**Example:**

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

department VARCHAR(50),

hire\_date DATE

);

**6. ALTER TABLE Statement**

Used to modify an existing table structure.

ALTER TABLE table\_name

ADD column\_name datatype;

**Example:**

ALTER TABLE employees

ADD email VARCHAR(100);

**7. DROP TABLE Statement**

Used to delete an entire table and its data.

DROP TABLE table\_name;

**Example:**

DROP TABLE employees;

**8. CREATE INDEX Statement**

Used to create an index on a table (improves query performance).

CREATE INDEX index\_name

ON table\_name (column\_name);

**Example:**

CREATE INDEX idx\_last\_name

ON employees (last\_name);

**9. JOIN Operations**

Used to combine rows from two or more tables based on a related column.

**INNER JOIN:**

SELECT columns

FROM table1

INNER JOIN table2 ON table1.common\_column = table2.common\_column;

INNER JOIN table3 ON table1.common\_column = table2.common\_column;

INNER JOIN table4 ON table1.common\_column = table2.common\_column;

**Example:**

SELECT employees.first\_name, employees.last\_name, departments.department\_name

FROM employees

INNER JOIN departments

ON employees.department = departments.department\_id;

**LEFT JOIN:**

SELECT columns

FROM table1

LEFT JOIN table2

ON table1.common\_column = table2.common\_column;

**Example:**

SELECT employees.first\_name, employees.last\_name, departments.department\_name

FROM employees

LEFT JOIN departments

ON employees.department = departments.department\_id;

**10. GROUP BY Statement**

Used to group rows that have the same values into summary rows.

SELECT column1, aggregate\_function(column2)

FROM table\_name

GROUP BY column1;

**Example:**

SELECT department, COUNT(\*)

FROM employees

GROUP BY department;

**11. HAVING Clause**

Used to filter groups of rows based on aggregate functions.

SELECT column1, aggregate\_function(column2)

FROM table\_name

GROUP BY column1

HAVING aggregate\_function(column2) condition;

**Example:**

SELECT department, COUNT(\*)

FROM employees

GROUP BY department

HAVING COUNT(\*) > 5;

**12. ORDER BY Clause**

Used to sort the result set.

SELECT column1, column2, ...

FROM table\_name

ORDER BY column1 [ASC|DESC];

**Example:**

SELECT first\_name, last\_name

FROM employees

ORDER BY last\_name ASC;

These are the core components of SQL syntax. Depending on the database system you are using (like MySQL, PostgreSQL, SQL Server, etc.), there might be some variations or additional features.

Certainly! Beyond the basics, SQL includes several more advanced syntax features and functionalities. Here’s a deeper dive into some additional SQL syntax and examples:

**13. Subqueries**

A subquery is a query within another query. It can be used in SELECT, INSERT, UPDATE, and DELETE statements.

**In SELECT:**

SELECT column1, column2

FROM table\_name

WHERE column1 = (SELECT column1 FROM other\_table WHERE condition);

**Example:**

SELECT first\_name, last\_name

FROM employees

WHERE department\_id = (SELECT department\_id FROM departments WHERE department\_name = 'Sales');

**In FROM:**

SELECT column1, column2

FROM (SELECT column1, column2 FROM table\_name WHERE condition) AS subquery

WHERE condition;

**Example:**

SELECT \*

FROM (SELECT first\_name, last\_name FROM employees WHERE department = 'Sales') AS sales\_employees

WHERE last\_name LIKE 'S%';

**14. UNION and UNION ALL**

Used to combine the results of two or more SELECT queries.

**UNION:**

Combines results and removes duplicates.

SELECT column1 FROM table1

UNION

SELECT column1 FROM table2;

**UNION ALL:**

Combines results including duplicates.

SELECT column1 FROM table1

UNION ALL

SELECT column1 FROM table2;

**Example:**

SELECT first\_name FROM employees

WHERE department = 'Sales'

UNION

SELECT first\_name FROM contractors

WHERE department = 'Sales';

**15. CASE Statement**

Used for conditional logic in queries.

SELECT column1,

CASE

WHEN condition1 THEN result1

WHEN condition2 THEN result2

ELSE default\_result

END AS alias\_name

FROM table\_name;

**Example:**

SELECT first\_name,

CASE

WHEN salary > 50000 THEN 'High'

WHEN salary BETWEEN 30000 AND 50000 THEN 'Medium'

ELSE 'Low'

END AS salary\_bracket

FROM employees;

**16. COALESCE Function**

Returns the first non-null value from the list of arguments.

SELECT COALESCE(column1, column2, 'default\_value') AS result

FROM table\_name;

**Example:**

SELECT COALESCE(email, 'No email provided') AS contact\_email

FROM employees;

**17. NULL Handling**

**IS NULL:**

SELECT column1

FROM table\_name

WHERE column1 IS NULL;

**IS NOT NULL:**

SELECT column1

FROM table\_name

WHERE column1 IS NOT NULL;

**Example:**

SELECT first\_name

FROM employees

WHERE email IS NULL;

**18. DISTINCT**

Used to remove duplicate rows from the result set.

SELECT DISTINCT column1

FROM table\_name;

**Example:**

SELECT DISTINCT department

FROM employees;

**19. LIMIT and OFFSET**

Control the number of rows returned by a query and skip a number of rows, respectively.

**LIMIT:**

SELECT column1

FROM table\_name

LIMIT number;

**OFFSET:**

SELECT column1

FROM table\_name

OFFSET number;

**Example:**

SELECT first\_name

FROM employees

ORDER BY last\_name

LIMIT 10 OFFSET 5;

**20. TRANSACTIONS**

**BEGIN TRANSACTION:**

Starts a new transaction.

BEGIN TRANSACTION;

**COMMIT:**

Saves the changes made during the transaction.

COMMIT;

**ROLLBACK:**

Undoes changes made during the transaction.

ROLLBACK;

**Example:**

BEGIN TRANSACTION;

UPDATE employees

SET department = 'HR'

WHERE department = 'Finance';

-- If everything is correct

COMMIT;

-- If there was an error

--ROLLBACK;

**21. Indexes**

**DROP INDEX:**

Deletes an index from a table.

DROP INDEX index\_name;

**Example:**

DROP INDEX idx\_last\_name;

**22. ALTER Table: Modifying Columns**

**Change Column Type:**

ALTER TABLE table\_name

ALTER COLUMN column\_name TYPE new\_data\_type;

**Rename Column:**

ALTER TABLE table\_name

RENAME COLUMN old\_column\_name TO new\_column\_name;

**Example:**

ALTER TABLE employees

ALTER COLUMN hire\_date TYPE TIMESTAMP;

ALTER TABLE employees

RENAME COLUMN hire\_date TO start\_date;

**23. VIEW**

A virtual table based on the result of a query.

**Create View:**

CREATE VIEW view\_name AS

SELECT column1, column2, ...

FROM table\_name

WHERE condition;

**Example:**

CREATE VIEW sales\_employees AS

SELECT first\_name, last\_name

FROM employees

WHERE department = 'Sales';

**Drop View:**

DROP VIEW view\_name;

**Example:**

DROP VIEW sales\_employees;

**24. Stored Procedures and Functions**

Used to encapsulate complex logic.

**Create Procedure:**

CREATE PROCEDURE procedure\_name AS

BEGIN

-- SQL statements

END;

**Create Function:**

CREATE FUNCTION function\_name (parameters)

RETURNS return\_data\_type AS

BEGIN

-- SQL statements

RETURN result;

END;

**Example (Procedure):**

CREATE PROCEDURE GetEmployeeDetails

AS

BEGIN

SELECT \* FROM employees;

END;

**Example (Function):**

CREATE FUNCTION GetEmployeeCount()

RETURNS INT AS

BEGIN

DECLARE @count INT;

SELECT @count = COUNT(\*) FROM employees;

RETURN @count;

END;