SQL (Structured Query Language) syntax is the set of rules that govern how you write SQL statements. Here's a breakdown of key aspects:

1. Basic Structure of a SQL Query (SELECT Statement):

- The most common SQL operation is retrieving data using the `SELECT` statement. The basic structure is:

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

- `SELECT`: Specifies the columns you want to retrieve.
- `FROM`: Specifies the table(s)[1] containing the data.
- `WHERE`: (Optional) Filters the rows based on a specified condition.
- `;`: Terminates the SQL statement (required in many SQL environments).

2. Key SQL Keywords:

- `SELECT`, `FROM`, `WHERE`, `INSERT`, `UPDATE`, `DELETE`, `CREATE`, `ALTER`, `DROP`, `JOIN`, `GROUP BY`, `HAVING`, `ORDER BY`, `LIMIT`, `OFFSET`, `DISTINCT`, `AS`.

3. Data Manipulation Language (DML):

- `INSERT`: Adds new rows to a table.
  - Example: `INSERT INTO Customers (CustomerID, CustomerName, City) VALUES (1, 'John Doe', 'New York');`
- `UPDATE`: Modifies existing rows in a table.
  - Example: `UPDATE Customers SET City = 'Los Angeles' WHERE CustomerID = 1;`
- `DELETE`: Removes rows from a table.
  - Example: `DELETE FROM Customers WHERE CustomerID = 1;`

4. Data Definition Language (DDL):

- `CREATE TABLE`: Creates a new table.
  - Example: `CREATE TABLE Employees (EmployeeID INT PRIMARY KEY, LastName VARCHAR(255), FirstName VARCHAR(255));`
- `ALTER TABLE`: Modifies an existing table (e.g., adds or removes columns).
  - Example: `ALTER TABLE Employees ADD COLUMN Department VARCHAR(255);`
- `DROP TABLE`: Deletes a table.
  - Example: `DROP TABLE Employees;`

5. Data Control Language (DCL):

- `GRANT`: Gives users permissions to access database objects.
  - Example: `GRANT SELECT ON Customers TO user1;`

- `REVOKE`: Removes permissions from users.
  - Example: `REVOKE SELECT ON Customers FROM user1;`

6. Clauses:

- `WHERE` Clause:
  - Used to filter rows based on conditions.
  - Operators: `=`, `!=` (or `<>`), `>`, `<`, `>=`, `<=`, `LIKE`, `IN`, `BETWEEN`, `AND`, `OR`, `NOT`.
- `ORDER BY` Clause:
  - Sorts the result set.
  - `ASC` (ascending) or `DESC` (descending).
  - Example: `SELECT * FROM Customers ORDER BY CustomerName ASC;`
- `GROUP BY` Clause:
  - Groups rows with the same values into summary rows.
  - Used with aggregate functions (e.g., `COUNT()`, `SUM()`, `AVG()`, `MAX()`, `MIN()`).
  - Example: `SELECT City, COUNT(CustomerID) FROM Customers GROUP BY City;`
- `HAVING` Clause:
  - Filters the results of a `GROUP BY` clause.
  - Example: `SELECT City, COUNT(CustomerID) FROM Customers GROUP BY City HAVING COUNT(CustomerID) > 2;`
- `JOIN` Clause:
  - Combines rows from two or more tables based on a related column.
  - Types: `INNER JOIN`, `LEFT JOIN`, `RIGHT JOIN`, `FULL OUTER JOIN`.
  - Example: `SELECT Customers.CustomerName, Orders.OrderID FROM Customers INNER JOIN Orders ON Customers.CustomerID = Orders.CustomerID;`[2]
- `LIMIT`/`OFFSET` Clauses:
  - Used to specify the number of records to return, and where to start returning records. These are very useful for pagination.
  - Example: `SELECT * FROM Customers LIMIT 10 OFFSET 20;` Returns 10 records, starting from record 21.

7. Aggregate Functions:

- `COUNT()`: Counts the number of rows.
- `SUM()`: Calculates the sum of a column.
- `AVG()`: Calculates the average of a column.
- `MAX()`: Finds the maximum value in a column.
- `MIN()`: Finds the minimum value in a column.[3]

8. Aliases:

- Used to give a table or column a temporary name using the `AS` keyword.
  - Example: `SELECT CustomerName AS Name FROM Customers;`

9. Wildcards:

- `%`: Represents zero or more characters (used with the `LIKE` operator).
- _: Represents a single character (used with the `LIKE` operator).

10. Data Types:

- `INT`, `VARCHAR`, `CHAR`, `DATE`, `DATETIME`, `DECIMAL`, `FLOAT`, `BOOLEAN`.

Important Notes:

- SQL syntax can vary slightly between different database systems (e.g., MySQL, PostgreSQL, SQL Server, Oracle).
- Case sensitivity can also vary; some systems are case-sensitive, and others are not. It is generally considered good practice to write SQL keywords in uppercase.
- Proper indentation and formatting make SQL queries more readable.