SQL-Structured Query Language

- SQL is a programming language used for querying and managing data in relational databases.
- SQL is the most widely implemented database language supported by popular relational database systems, like MySQL, SQL Server, and Oracle.
- MySQL is a relational database management system(RDBMs) developed by Oracle that is based on structured query language(SQL).

Applications Of SQL:

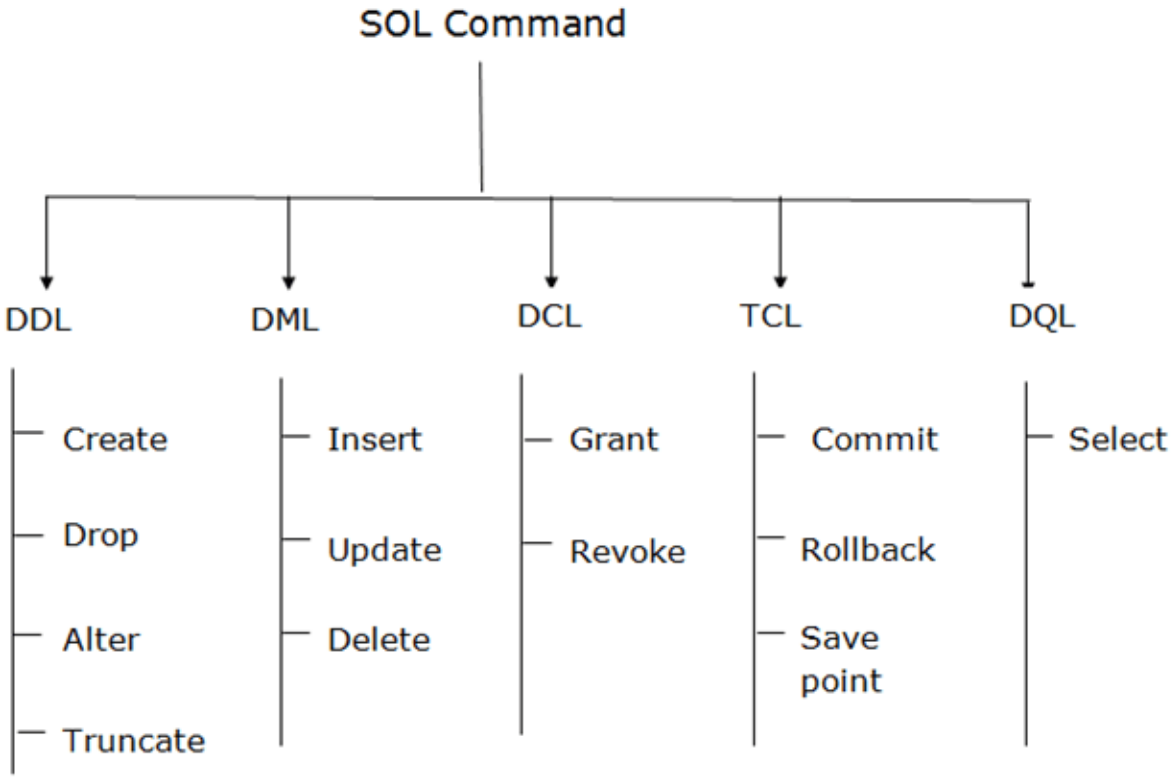
- **Data Integration Scripts:**
  The main application of SQL is to write data integration scripts by the database administrators and developers.
- **Retrieve Information:**
  It retrieve the subsets of information within a database for analytics applications and transaction processing. The most commonly used SQL elements are SELECT, INSERT, UPDATE, ADD, DELETE, CREATE, TRUNCATE and ALTER.
- **Other Important Applications:**
  The SQL is used for modification of index structures and database table. Additionally, the users can ADD, UPDATE and DELETE the rows of the data by using this language.

Advantages:

- Large amount of data is retrieved quickly and efficiently.
- Operations like INSERTION, DELETION, MANIPULATION of data is also done in almost no time.
- Easy to learn and understand, answers to complex queries can be received in seconds.

Disadvantages:

- SQL has a difficult interface that makes few users uncomfortable while dealing with the database.
- Some versions are costly and hence, programmers cannot access it.
- Due to hidden business rules, complete control is not given to the database.



DDL(DATA DEFINITION LANGUAGE):

- It is used to define the database structure or schema. DDL is also used to specify additional properties of data.
    1. CREATE: to create objects in database.
    2. ALTER: alters the structure of database.
    3. DROP: delete objects from database.
    4. RENAME: rename an object.

DQL(DATA QUERY LANGUAGE):

- It statements are used for performing queries on the data within schema objects. The purpose of the DQL Command is to get some schema relation based on the query passed to it.
    1. SELECT: it is used to retrieve data from the database.

DML(DATA MANIPULATION LANGUAGE):

- DML statements are used for managing data with in schema objects.
- DML are of two types,
    1. **Procedural DMLs**:
       It require a user to specify what data are needed and how to get those data.
    2. **Declarative DMLs(also referred as Non-Procedural DMLs)**:
       It require a user to specify what data are needed without specifying how to get those data.

- SELECT: retrieve data from the database.
- INSERT: insert data into a table
- UPDATE: update existing data within a table.
- DELETE: deletes all records from a table, space for the record remain.

DCL(DATA CONTROL LANGUAGE):

- A DCL is a system is a syntax similar to a computer programming language used to control access to data stored in a database.
    - GRANT: allow specified user to perform specified tasks.
    - REVOKE: cancel previously granted or denied permission.

TCL(TRANSACTION CONTROL LANGUAGE):

- TCL commands are used to manage transactions in the database. These are used to manage the changes made by DML-statements.
- It allows statements to be grouped together into logical transaction.
    - COMMIT: It is used to permanently save any transaction into the database.
    - ROLLBACK: This command restores the database to last committed state.
    - It also used with save point command to jump to a save point in a transaction.
    - SAVEPOINT: This command is used to temporarily save a transaction so that you can roll back to that point whenever necessary.

What is DATABASE:

- A database is a systematic collection of data.
- They support electronic storage and manipulation of data.
- Databases make data management easy.
- A database is an organized collection of structured information or data.

DBMS:

- DBMS are software systems used to store, retrieve, and run queries on data.
- CRUD[CREATE, READ, UPDATE, DELETE]

Types of DBMS:

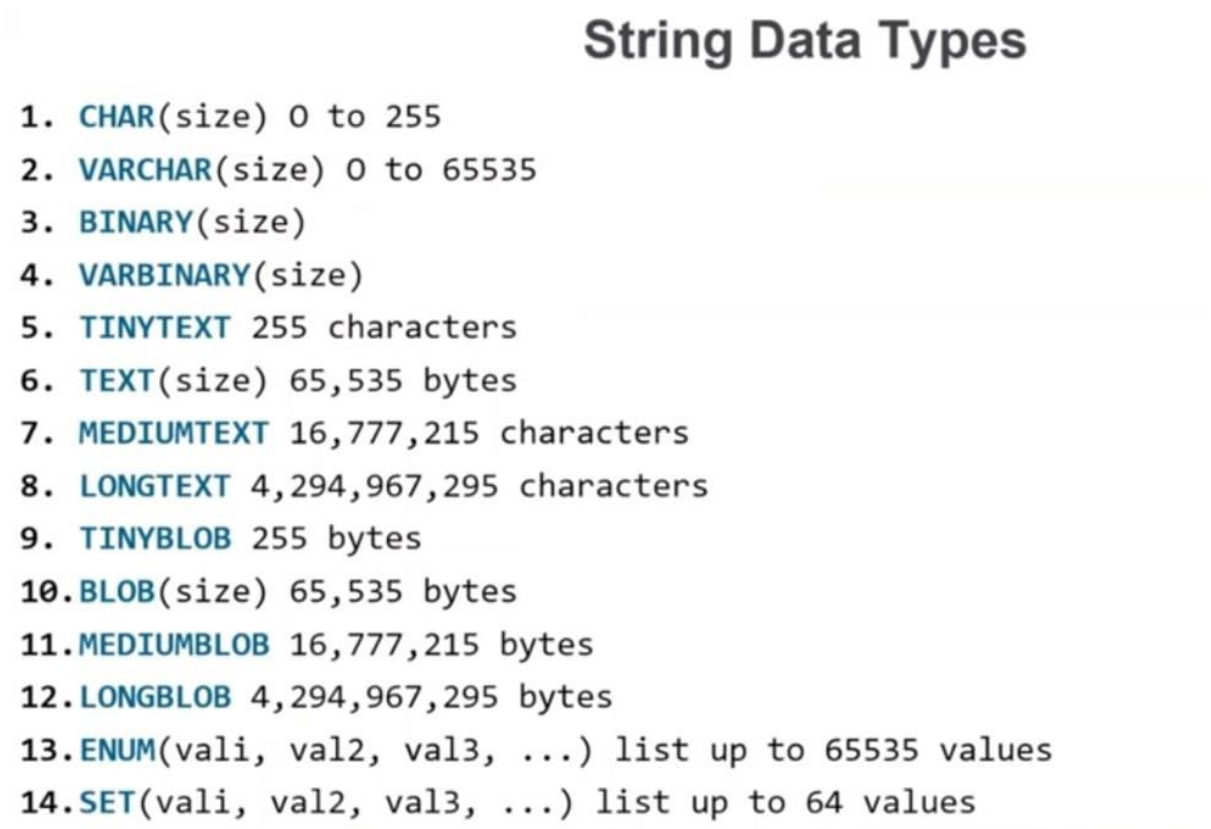
1. Relational DBMS (SQL)
2. Non-Relational DBMS (No SQL)

RDBMS:

- RDBMS stands for RELATIONAL DATABASE MANAGEMENT SYSTEM.
- It is the basis for SQL, and for all modern database systems such as MS SQL Server, IBM DB2, ORACLE, MYSQL, and MICROSOFT ACCESS.
- The data in RDBMS is stored in database objects called TABLES.
- A table is a collection of related data entries and it consists of columns and rows.

- A relational database is a collection of data items with pre-defined relationships between them, stored in the form of table, rows and columns.
- List of SQL databases,
  MySQL, MariaDB, Oracle, PostgreSQL, MS SQL

Categories of Data Types:



Number Data Types

1. BIT(size) 1 to 64
2. TINYINT(size) -128 to 127
3. INT(size) -2147483648 to 2147483647
4. INTEGER(size)
5. SMALLINT(size) -32768 to 32767
6. MEDIUMINT(size) -8388608 to 8388607
7. BIGINT(size) -9223372036854775808 to 9223372036854775
8. BOOL
9. BOOLEAN 0/1
10. FLOAT(p)
11. DOUBLE(size, d) 255.568
12. DECIMAL(size, d) Size = 60, d = 30
13. DEC(size, d)

Date Data Types

1. DATE '1000-01-01' to '9999-12-31’
2. DATETIME(fsp) YYYY-MM-DD hh:mm:ss
3. TIMESTAMP(fsp)
4. TIME(fsp) hh:mm:ss
5. YEAR four-digit format : 1901

Most Frequently Used Data Types:

Numeric data types in SQL

Data Type	From	To
bigint	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
int	-2,147,483,648	2,147,483,647
smallint	-32,768	32,767
tinyint	0	255
Bit	0	1
decimal	-10^38 +1	10^38 -1
money	-922,337,203,685,477.5808	922,337,203,685,477.5807
float	-1.79E + 308	1.79E + 308

Date and Time Data Types

Data Type	From	To
datetime	Jan 1, 1753 with time	Dec 31, 9999 with time
smalldatetime	Jan 1, 1900	Jun 6, 2079
Date	Jan 1, 1753	Dec 31, 9999

Character Strings Data Types

- Char : Maximum length of 8,000 characters.(Fixed length non-Unicode characters)
- varchar: Maximum of 8,000 characters. (Variable-length non-Unicode data).
- varchar(max) : Maximum length of 2E + 31 characters, Variable-length non-Unicode data (SQL Server 2005 only).
- Text : Variable-length non-Unicode data with a maximum length of 2,147,483,647 characters

QUERY:

- SQL queries are made up of commands that allow you to manipulate data within a database. These commands follow a specific syntax (a set of rules) so that they're interpreted correctly by the database management system (DBMS).
- A query can either be a request for data results from your database or for action on the data, or for both.
- A query can give you an answer to a simple question, perform calculations, combine data from different tables, add, change, or delete data from a database.
- Query➔Codes

❖ ***-Relationship➔ How two columns are connected to each other.***

❖ ***-Blue Print of database is called Schema.***

Table Constraints:

LIKE Operator	Description
LIKE 'a%'	Starts with "a"
LIKE '%a'	End with "a"
LIKE '%or%'	Have "or" in any position
LIKE '_r%'	Have "r" in the second position
LIKE 'a_%'	Starts with "a" and are at least 2 characters in length
LIKE 'a__%'	Starts with "a" and are at least 3 characters in length
LIKE 'a%o'	Starts with "a" and ends with "o"

Semicolon after SQL Statements?

Some database systems require a semicolon at the end of each SQL statement.

Semicolon is the standard way to separate each SQL statement in database systems that allow more than one SQL statement to be executed in the same call to the server.

Some of The Most Important SQL Commands:

- SELECT - extracts data from a database
- UPDATE - updates data in a database
- DELETE - deletes data from a database
- INSERT INTO - inserts new data into a database
- CREATE DATABASE - creates a new database
- ALTER DATABASE - modifies a database
- CREATE TABLE - creates a new table
- ALTER TABLE - modifies a table
- DROP TABLE - deletes a table
- CREATE INDEX - creates an index (search key)
- DROP INDEX - deletes an index

SQL SELECT Statement:

The SELECT statement is used to select data from a database.

The data returned is stored in a result table, called the result-set.

SELECT Syntax:

SELECT *column1*, *column2, ...*
FROM *table_name*;

If you want to select all the fields available in the table, use the following syntax:

SELECT * FROM *table_name*;

SQL SELECT DISTINCT Statement:

The SELECT DISTINCT statement is used to return only distinct (different) values.

Inside a table, a column often contains many duplicate values; and sometimes you only want to list the different (distinct) values.

SELECT DISTINCT Syntax:

SELECT DISTINCT *column1*, *column2, ...*
FROM *table_name*;

SQL WHERE Clause

The WHERE clause is used to filter records.

It is used to extract only those records that fulfill a specified condition.

WHERE Syntax:

SELECT *column1*, *column2, ...*
FROM *table_name*
WHERE *condition*;

Note: The WHERE clause is not only used in SELECT statements, it is also used in UPDATE, DELETE, etc.!

Operator	Description
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal. **Note:** In some versions of SQL this operator may be written as !=
BETWEEN	Between a certain range
LIKE	Search for a pattern
IN	To specify multiple possible values for a column

SQL AND, OR and NOT Operators:

The WHERE clause can be combined with AND, OR, and NOT operators.

The AND and OR operators are used to filter records based on more than one condition:

- The AND operator displays a record if all the conditions separated by AND are TRUE.

- The OR operator displays a record if any of the conditions separated by OR is TRUE.

The NOT operator displays a record if the condition(s) is NOT TRUE.

AND Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 AND condition2 AND condition3 ...;
```

OR Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE condition1 OR condition2 OR condition3 ...;
```

NOT Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE NOT condition;
```

SQL ORDER BY:

The ORDER BY keyword is used to sort the result-set in ascending or descending order.

The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

ORDER BY Syntax:

```
SELECT column1, column2, ...
FROM table_name
ORDER BY column1, column2, ... ASC|DESC;
```

SQL INSERT INTO Statement:

The INSERT INTO statement is used to insert new records in a table.

INSERT INTO Syntax:

It is possible to write the INSERT INTO statement in two ways:

1. Specify both the column names and the values to be inserted:

INSERT INTO *table_name* (*column1*, *column2*, *column3*, ...)
VALUES (*value1*, *value2*, *value3*, ...);

2. If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. Here, the INSERT INTO syntax would be as follows:

INSERT INTO *table_name*
VALUES (*value1*, *value2*, *value3*, ...);

What is a NULL Value?

A field with a NULL value is a field with no value.

If a field in a table is optional, it is possible to insert a new record or update a record without adding a value to this field. Then, the field will be saved with a NULL value.

Note: A NULL value is different from a zero value or a field that contains spaces. A field with a NULL value is one that has been left blank during record creation!

How to Test for NULL Values?

It is not possible to test for NULL values with comparison operators, such as =, <, or <>.

We will have to use the IS NULL and IS NOT NULL operators instead.

IS NULL Syntax:

SELECT *column_names*
FROM *table_name*
WHERE *column_name* IS NULL;

IS NOT NULL Syntax:

SELECT *column_names*
FROM *table_name*
WHERE *column_name* IS NOT NULL;

SQL UPDATE Statement:

The UPDATE statement is used to modify the existing records in a table.

UPDATE Syntax:

UPDATE *table_name*
SET *column1* = *value1*, *column2* = *value2*, ...
WHERE *condition*;

Note: Be careful when updating records in a table! Notice the WHERE clause in the UPDATE statement. The WHERE clause specifies which record(s) that should be updated. If you omit the WHERE clause, all records in the table will be updated!

SQL DELETE Statement:

The DELETE statement is used to delete existing records in a table.

DELETE Syntax:

DELETE FROM *table_name* WHERE *condition*;

Note: Be careful when deleting records in a table! Notice the WHERE clause in the DELETE statement. The WHERE clause specifies which record(s) should be deleted. If you omit the WHERE clause, all records in the table will be deleted!

SQL MIN() and MAX() Functions:

The MIN() function returns the smallest value of the selected column.

The MAX() function returns the largest value of the selected column.

MIN() Syntax:

SELECT MIN(*column_name*)
FROM *table_name*
WHERE *condition*;

MAX() Syntax:

SELECT MAX(*column_name*)
FROM *table_name*
WHERE *condition*;

SQL COUNT(), AVG() and SUM() Functions:

The COUNT() function returns the number of rows that matches a specified criterion.

COUNT() Syntax:

```
SELECT COUNT(column_name)
FROM table_name
WHERE condition;
```

The AVG() function returns the average value of a numeric column.

AVG() Syntax:

```
SELECT AVG(column_name)
FROM table_name
WHERE condition;
```

The SUM() function returns the total sum of a numeric column.

SUM() Syntax:

```
SELECT SUM(column_name)
FROM table_name
WHERE condition;
```

SQL LIKE Operator:

The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

There are two wildcards often used in conjunction with the LIKE operator:

- The percent sign (%) represents zero, one, or multiple characters
- The underscore sign (_) represents one, single character

Note: MS Access uses an asterisk (*) instead of the percent sign (%), and a question mark (?) instead of the underscore (_).

The percent sign and the underscore can also be used in combinations!

LIKE Syntax:

```
SELECT column1, column2, ...
FROM table_name
WHERE columnN LIKE pattern;
```

Tip: You can also combine any number of conditions using AND or OR operators.

SQL Wildcard Characters

- A wildcard character is used to substitute one or more characters in a string.
- Wildcard characters are used with the LIKE operator. The LIKE operator is used in a WHERE clause to search for a specified pattern in a column.

SQL IN Operator:

- The IN operator allows you to specify multiple values in a WHERE clause.
- The IN operator is a shorthand for multiple OR conditions.

IN Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (value1, value2, ...);
```

or:

```
SELECT column_name(s)
FROM table_name
WHERE column_name IN (SELECT STATEMENT);
```

SQL BETWEEN Operator:

- The BETWEEN operator selects values within a given range. The values can be numbers, text, or dates.
- The BETWEEN operator is inclusive: begin and end values are included.

BETWEEN Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE column_name BETWEEN value1 AND value2;
```

SQL Aliases:

- SQL aliases are used to give a table, or a column in a table, a temporary name.
- Aliases are often used to make column names more readable.
- An alias only exists for the duration of that query.
- An alias is created with the AS keyword.

Alias Column Syntax:

SELECT *column_name* AS *alias_name*
FROM *table_name;*

Alias Table Syntax:

SELECT *column_name(s)*
FROM *table_name* AS *alias_name;*

SQL GROUP BY Statement:

The GROUP BY statement groups rows that have the same values into summary rows, like "find the number of customers in each country".

The GROUP BY statement is often used with aggregate functions (COUNT(), MAX(), MIN(), SUM(), AVG()) to group the result-set by one or more columns.

GROUP BY Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
ORDER BY column_name(s);
```

SQL HAVING Clause:

The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions.

HAVING Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE condition
GROUP BY column_name(s)
HAVING condition
ORDER BY column_name(s);
```

SQL EXISTS Operator:

The EXISTS operator is used to test for the existence of any record in a subquery.

The EXISTS operator returns TRUE if the subquery returns one or more records.

EXISTS Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE EXISTS
(SELECT column_name FROM table_name WHERE condition)
```

SQL ANY and ALL Operators:

The ANY and ALL operators allow you to perform a comparison between a single column value and a range of other values.

SQL ANY Operator:

The ANY operator:

- returns a boolean value as a result
- returns TRUE if ANY of the subquery values meet the condition

ANY means that the condition will be true if the operation is true for any of the values in the range.

ANY Syntax:

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator ANY
  (SELECT column_name
  FROM table_name
  WHERE condition);
```

Note: The *operator* must be a standard comparison operator (=, <>, !=, >, >=, <, or <=).

SQL ALL Operator:

The ALL operator:

- returns a boolean value as a result
- returns TRUE if ALL of the subquery values meet the condition
- is used with SELECT, WHERE and HAVING statements

ALL means that the condition will be true only if the operation is true for all values in the range.

ALL Syntax With SELECT:

```
SELECT ALL column_name(s)
FROM table_name
WHERE condition;
```

ALL Syntax With WHERE or HAVING:

```
SELECT column_name(s)
FROM table_name
WHERE column_name operator ALL
  (SELECT column_name
  FROM table_name
  WHERE condition);
```

Note: The *operator* must be a standard comparison operator (=, <>, !=, >, >=, <, or <=).

SQL SELECT INTO Statement:

The SELECT INTO statement copies data from one table into a new table.

SELECT INTO Syntax:

Copy all columns into a new table:

```
SELECT *
INTO newtable [IN externaldb]
FROM oldtable
WHERE condition;
```

Copy only some columns into a new table:

```
SELECT column1, column2, column3, ...
INTO newtable [IN externaldb]
FROM oldtable
WHERE condition;
```

The new table will be created with the column-names and types as defined in the old table. You can create new column names using the AS clause

SQL INSERT INTO SELECT Statement:

The INSERT INTO SELECT statement copies data from one table and inserts it into another table.

The INSERT INTO SELECT statement requires that the data types in source and target tables match.

Note: The existing records in the target table are unaffected.

INSERT INTO SELECT Syntax:

Copy all columns from one table to another table:

```
INSERT INTO table2
SELECT * FROM table1
WHERE condition;
```

Copy only some columns from one table into another table:

```
INSERT INTO table2 (column1, column2, column3, ...)
SELECT column1, column2, column3, ...
FROM table1
WHERE condition;
```