**Numeric Types:**

**TINYINT :** TINYINT is a numeric data type that represents a small integer value.

It can store integer values ranging from -128 to 127 for signed TINYINT or 0 to 255 for unsigned TINYINT.

* Size: TINYINT occupies 1 byte of storage.
* Range: For signed TINYINT, the range is from -128 to 127. For unsigned TINYINT, the range is from 0 to 255.
* Default: The default value for TINYINT is 0.
* Usage: TINYINT is commonly used when you need to store small integer values, such as 0 for inactive and 1 for active

**INT:** Signed integer values (whole numbers).

* Size: The INT data type occupies 4 bytes of storage.
* Range: The signed INT can store values from -2,147,483,648 to 2,147,483,647, while the unsigned INT can store values from 0 to 4,294,967,295.
* Default: The default value for an INT column is 0.
* Usage: INT is commonly used for storing numeric values such as IDs, counts, quantities, and other whole numbers that may have a wide range of values.

**BIGINT:** Larger signed integer values.

* Size: The BIGINT data type occupies 8 bytes of storage.
* Range: The signed BIGINT can store values from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807, while the unsigned BIGINT can store values from 0 to 18,446,744,073,709,551,615.
* Default: The default value for a BIGINT column is 0.
* Usage: BIGINT is typically used for storing very large numeric values, such as unique identifiers, counters, or other numbers that exceed the range of the INT data type.

**DECIMAL:** Fixed-point decimal numbers.

Precision: The precision specifies the total number of digits (both before and after the decimal point) that can be stored in the column.

**FLOAT:** Single-precision floating-point numbers.

Precision: The precision of a FLOAT value is approximately 7 digits.

**DOUBLE: Double-precision floating-point numbers.**

Precision: The precision of a DOUBLE value is approximately 15 digits.

**String Types:**

**VARCHAR:** The VARCHAR data type is used to store variable-length character strings. It is commonly used for storing textual data such as names, addresses, descriptions

* Size: The VARCHAR data type requires storage based on the length of the string being stored. It can store strings with a maximum length of up to 65,535 characters (bytes).
* Length: When defining a VARCHAR column, you need to specify the maximum length for the column. For example, VARCHAR(100) indicates that the column can store strings up to 100 characters in length.
* Storage: VARCHAR only occupies the actual storage needed for the stored string, plus 1 or 2 bytes to store the length of the string.
* Default: The default value for a VARCHAR column is NULL, unless specified otherwise.

**CHAR:** Fixed-length character strings.

* Size: The CHAR data type requires storage based on the defined length. It always occupies a fixed amount of storage equal to the specified length multiplied by the number of bytes required for the character set.
* Length: When defining a CHAR column, you need to specify the fixed length for the column. For example, CHAR(10) indicates that the column will store strings of exactly 10 characters in length.
* Storage: CHAR always occupies the full defined length of the column, regardless of the actual length of the stored string.
* Padding: If the length of the stored string is less than the defined length, MySQL pads the remaining spaces with blank characters.

**TEXT:** The TEXT data type is used to store large blocks of text data, such as long descriptions, comments, or textual content that can vary in length. ENUM: A set of predefined

* Size: The TEXT data type can store up to 65,535 bytes (characters) of text data.
* Length: Unlike other data types like VARCHAR or CHAR, the TEXT data type does not require specifying a length limit.
* Storage: TEXT data is stored separately from the table structure and can take up additional space based on the actual length of the stored text.

**Enum:** The ENUM data type is used to define a column that can store one value from a predefined list of possible values. It allows you to define a set of allowed values for a column and enforce data integrity by ensuring that only those specified values can be inserted or updated.

**Date and Time Types:**

**DATE:** Stores a date value in the format 'YYYY-MM-DD'. It represents a date without a specific time of day.

**TIME:** Stores a time value in the format 'HH:MM:SS'. It represents a time of day without a specific date.

**DATETIME:** Stores both date and time values in the format 'YYYY-MM-DD HH:MM:SS'. It represents a specific date and time.

**TIMESTAMP:** Stores a timestamp value representing the number of seconds since the Unix epoch (1970-01-01 00:00:00 UTC). It can store a range of dates from 1970 to 2038.

**YEAR**: Stores a year value in either a four-digit or two-digit format. The values range from 1901 to 2155 (or 1970 to 2069 for two-digit format).

**Difference between Char and Varchar in SQL**

**(Both are used to store the characters)**

**Varchar** - Variable Length

**Char** - Fixed Length

Ex:- Char (20) ------- (Jeevan Raj\_\_\_\_\_\_\_\_\_\_) ……(it take 20 bites no matters what is the length of your string)

Ex:- Varchar(20) ----- (Jeevan Raj) ……(It takes only the 10 bites, It not takes entire memory)

|  |  |
| --- | --- |
| Char | Varchar |
| Uses Static Memory Allocation | It uses Dynamic Memory Allocation |
| Fixed Length | Variable Length (Better for storage space) |
| Faster | Slower  (disadvantage:- Has an overhead of 2 bites extra) |

Use case of Char:- Phone Number, Zip Code etc:-

Use case of Varchar :- Names, Address, Location