**Understanding Data Types**

Understanding the data types is important because storing data in the appropriate format is fundamental to building usable databases and performing accurate analysis.

Data types that are encountered the most:

* Characters (Any character or symbol)
* Numbers
* Dates and times

**Characters:**

* Char(n) : A fixed-length column where the character length is specified by n. If n = 20, and you enter a value with less than 20 characters, SQL will still add padding in the form of spaces and have 20 characters in it.
* Varchar(n): A variable-length column where the maximum length is specified by n. SQL will not try to add more padding if the number of characters is less than n.
* Text: A variable length column of unlimited length

**Numbers:**

Note that numbers stored in the form of text, sort differently compared to normal numbers, floats etc.

* Integers: Whole numbers including both – and + and 0. SQL provides 3 integer types: smallint, integer and bigint. Small int is of 2 bytes, integer is 4 bytes, and bigint is 8 bytes.
* AutoIncrementing Integers: bigserial, smallserial and serial. These are not true datatypes, but special implementation of the corresponding integer type.

In autoincrementing integers, if a row is deleted that value of the row is never replaced. Or if a row is aborted, the sequence still increments.

**Decimal Numbers:**

We can use fixed-point or floating-point numbers.

* Fixed-point numbers: We define it as *demical(precision,scale)* or *numeric(precision,scale)* where precision is the number of digits allowable on both left and right of the decimal, and scale is the number of digits on the right of the decimal. Setting scale to 0 or omitting it, makes it to an integer.
* Floating point numbers: Real has 6 decimal precision, double precision has 15 decimal precision.

Floating point numbers(real and double) tends to give non exact answers as the computer tries to squeeze lots of information into a finite number of bits. The storage required by numeric data types tends to be variable and depending on he precision and scale specified, numeric can consume considerably more space than the floating point types. **However when dealing with tasks requiring exact values, use numeric(precision,scale)**