This page describes the data types that Cloud Firestore supports.

The following table lists the data types supported by Cloud Firestore. It also describes the sort order used when comparing values of the same type:

Data type	Sort order	Notes
Array	By element values	An array cannot contain another array value as one of its elements. Within an array, elements maintain the position assigned to them. When sorting tw arrays, arrays are ordered based on their element values. When comparing two arrays, the first elements of each array are compared. If the elements are equal, then the second elements are compared and so on until a diffe found. If an array runs out of elements to compare but is equal up to that point, th shorter array is ordered before the longer array. For example, `[1, 2, 3] < [1, 2, 3, 1] < [2]`. The array `[2]` has the greate element value. The array `[1, 2, 3]` has elements equal to the first three element `2, 3, 1]` but is shorter in length.
Boolean	`false` < `true`	—
Bytes	Byte order	Up to 1,048,487 bytes (1 MiB - 89 bytes). Only the first 1,500 bytes are considered
Date and time	Chronological	When stored in Cloud Firestore, precise only to microseconds; any additional precis rounded down.
Floating-point number	Numeric	64-bit double precision, IEEE 754.
Geographical point	By latitude, then longitude	—
Integer	Numeric	64-bit, signed

Data type	Sort order	Notes

Map	By keys, then by value	Represents an object embedded within a document. When indexed, you can query subfields. If you exclude this value from indexing, then all subfields are also exclud indexing.
		Key ordering is always sorted. For example, if you write `{c: "foo", a: "bar", "qux"}` the map is sorted by key and saved as `{a: "foo", b: "bar", c: "qu`
		Map fields are sorted by key and compared by key-value pairs, first comparing the then the values. If the first key-value pairs are equal, the next key-value pairs are co and so on. If two maps start with the same key-value pairs, then map length is cor For example, the following maps are in ascending order:
		`{a: "aaa", b: "baz"}`
		`{a: "foo", b: "bar"}`
		`{a: "foo", b: "bar", c: "qux"}`
		`{a: "foo", b: "baz"}`
		`{b: "aaa", c: "baz"}`
		`{c: "aaa"}`

Null	None	—
Reference	By path elements (collection, document ID, collection, document ID...)	For example, `projects/[PROJECT_ID]/databases/[DATABASE_ID]/documents/[DOCUME`
Text string	UTF-8 encoded byte order	Up to 1,048,487 bytes (1 MiB - 89 bytes). Only the first 1,500 bytes of the UTF-8 representation are considered by queries.

When a query involves a field with values of mixed types, Cloud Firestore uses a deterministic ordering based on the internal representations. The following list shows the order:

1. Null values
2. Boolean values

3. Integer and floating-point values, sorted in numerical order
4. Date values
5. Text string values
6. Byte values
7. Cloud Firestore references
8. Geographical point values
9. Array values
10. Map values