```swift
import Combine
import Dispatch
import Foundation
import _Concurrency
import _StringProcessing
import _SwiftConcurrencyShims

/// Generates a column by multiplying each element in an optional column type
/// by the corresponding elements of a column type.
/// - Parameters:
///   - lhs: An optional column type.
///   - rhs: A column type.
/// - Returns: A new column with the same type as the right column.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public func    L  R       L       R     Column R Element
where L   OptionalColumnProtocol  R   ColumnProtocol
L WrappedElement   Numeric  L WrappedElement    R Element

/// Generates a column by multiplying each element in a column type
/// by the corresponding elements of an optional column type.
/// - Parameters:
///   - lhs: A column type.
///   - rhs: An optional column type.
/// - Returns: A new column with the same type as the left column.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public func    L  R       L       R     Column L Element
where L   ColumnProtocol  R   OptionalColumnProtocol
L Element   Numeric  L Element    R WrappedElement

/// Generates a column by adding each element in an optional column type to the
corresponding elements of a column type.
/// - Parameters:
///   - lhs: An optional column type.
///   - rhs: A column type.
/// - Returns: A new column with the same type as the right column.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public func    L  R       L       R     Column R Element
where L   OptionalColumnProtocol  R   ColumnProtocol
L WrappedElement   AdditiveArithmetic  L WrappedElement
R Element

/// Generates a column by adding each element in a column type to the
corresponding elements of an optional column type.
/// - Parameters:
///   - lhs: A column type.
///   - rhs: An optional column type.
/// - Returns: A new column with the same type as the left column.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public func    L  R       L       R     Column L Element
where L   ColumnProtocol  R   OptionalColumnProtocol
```

```
L Element   AdditiveArithmetic  L Element    R WrappedElement

/// Generates a column by subtracting each element in a column type
/// from the corresponding elements of an optional column.
/// - Parameters:
///   - lhs: An optional column type.
///   - rhs: A column type.
/// - Returns: A new column with the same type as the right column.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public func    L  R       L       R     Column R Element
where L   OptionalColumnProtocol  R   ColumnProtocol
L WrappedElement   AdditiveArithmetic  L WrappedElement
R Element

/// Generates a column by subtracting each element in an optional column type
/// from the corresponding elements of a column type.
/// - Parameters:
///   - lhs: A column type.
///   - rhs: An optional column type.
/// - Returns: A new column with the same type as the left column.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public func    L  R       L       R     Column L Element
where L   ColumnProtocol  R   OptionalColumnProtocol
L Element   AdditiveArithmetic  L Element    R WrappedElement

/// Generates an integer column by dividing each element in an optional column
type
/// by the corresponding elements of a column type.
/// - Parameters:
///   - lhs: An optional column type.
///   - rhs: A column type.
/// - Returns: A new column with the same type as the right column.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public func    L  R       L       R     Column R Element
where L   OptionalColumnProtocol  R   ColumnProtocol
L WrappedElement   BinaryInteger  L WrappedElement
R Element

/// Generates an integer column by dividing each element in a column type
/// by the corresponding elements of an optional column type.
/// - Parameters:
///   - lhs: A column type.
///   - rhs: An optional column type.
/// - Returns: A new column with the same type as the left column.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public func    L  R       L       R     Column L Element
where L   ColumnProtocol  R   OptionalColumnProtocol
L Element   BinaryInteger  L Element    R WrappedElement
```

```
/// Generates a floating-point column by dividing each element in an optional column type
/// by the corresponding elements of a column type.
/// - Parameters:
///   - lhs: An optional column type.
///   - rhs: A column type.
/// - Returns: A new column with the same type as the right column.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public func    L  R       L       R     Column R Element
where L   OptionalColumnProtocol  R   ColumnProtocol
L WrappedElement   FloatingPoint  L WrappedElement
R Element

/// Generates a floating-point column by dividing each element in a column type
/// by the corresponding elements of an optional column type.
/// - Parameters:
///   - lhs: A column type.
///   - rhs: An optional column type.
/// - Returns: A new column with the same type as the left column.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public func    L  R       L       R     Column L Element
where L   ColumnProtocol  R   OptionalColumnProtocol
L Element   FloatingPoint  L Element    R WrappedElement

/// A type-erased categorical summary.
///
/// Categorical summary includes 5 statistics:
///   - someCount: The number of non-missing elements.
///   - noneCount: The number of missing elements.
///   - totalCount: The sum of missing and non-missing elements.
///   - uniqueCount: The number of unique elements.
///   - mode: The most common elements.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public struct AnyCategoricalSummary   Equatable

    /// The number of non-missing elements in the column.
    public var someCount  Int

    /// The number of missing elements in the column.
    public var noneCount  Int

    /// The total number of elements in the column.
    public var totalCount  Int   get

    /// The number of unique elements.
    public var uniqueCount  Int

    /// The most common values in a column.
    ///
```

```
    /// The summary orders the elements based on their original locations within the column.
    public var mode    Any

    /// The underlying type of ``mode``.
    public var modeType  any Any

    /// Creates a type-erased categorical summary from a typed categorical summary.
    ///
    /// - Parameters:
    ///   - summary: A typed categorical summary.
    public init T  _          CategoricalSummary T   where T Hashable

    /// Creates a type-erased categorical summary from a typed categorical summary.
    ///
    /// - Parameters:
    ///   - summary: A typed categorical summary.
    public init _          CategoricalSummary AnyHashable

    /// Returns a Boolean value that indicates whether the categorical summaries are equal.
    /// - Parameters:
    ///   - lhs: A type-erased categorical summary.
    ///   - rhs: Another type-erased categorical summary.
    public static func          AnyCategoricalSummary AnyCategoricalSummary     Bool


/// A type-erased column.
///
/// `AnyColumn` is a column type that conceals the type of its elements,
/// unlike ``Column``, its typed counterpart.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public struct AnyColumn   AnyColumnProtocol  Hashable

    /// The name of the column.
    public var name  String

    /// The underlying type of the column's elements.
    public var wrappedElementType  any Any        get

    /// A prototype that creates type-erased columns with the same underlying type as the column slice.
    ///
    /// Use a type-erased column prototype to create new columns of the same type as the slice's parent column
    /// without explicitly knowing what type it is, by calling
```

```
    /// the `prototype` property's
``AnyColumnPrototype/makeColumn(capacity:)`` method.
    ///
    /// ```swift
    /// // Get a type-erased column.
    /// let someColumn: AnyColumn = dataFrame["someFeature"]
    ///
    /// // Create a new column with the same type.
    /// let newColumn =
someColumn.prototype.makeColumn(capacity: 10)
    /// ```
    public var prototype  any AnyColumnPrototype   get

    /// The number of elements in the column.
    public var count  Int   get

    /// The number of missing elements in the column.
    @available macOS 12  iOS 15  tvOS 15  watchOS 8
    public var missingCount  Int   get

    /// Returns the underlying typed column.
    ///
    /// When using this method, you must provide the correct underlying type.
    ///
    /// - Parameter type: The type of the underlying column.
    /// - Returns: A typed column.
    public func assumingType T  _       T          Column T

    /// Returns a Boolean that indicates whether the element at the index is
missing.
    ///
    /// - Parameter index: The location of an element in the column.
    public func isNil           Int     Bool

    /// Appends an optional element to the column.
    ///
    /// - Parameter element: An element.
    public mutating func append _          Any

    /// Appends the contents of another column to the column.
    ///
    /// - Parameter other: A column that contains elements of the same
type as this column.
    public mutating func append                    AnyColumn

    /// Appends the contents of a column slice to the column.
    ///
    /// - Parameter other: A column that contains elements of the same
type as this column.
    public mutating func append
```

```
AnyColumnSlice

    /// Removes an element from the column.
    ///
    /// - Parameter index: The location of an element in the column.
    public mutating func remove            Int

    /// The hash value.
    ///
    /// Hash values are not guaranteed to be equal across different executions of
    /// your program. Do not save hash values to use during a future execution.
    ///
    /// - Important: `hashValue` is deprecated as a `Hashable`
requirement. To
    ///     conform to `Hashable`, implement the `hash(into:)` requirement
instead.
    ///     The compiler provides an implementation for `hashValue` for you.
    public var hashValue  Int   get


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension AnyColumn

    /// Decodes data for each element of the column.
    ///
    /// - Parameters:
    ///   - type: The type of the value to decode.
    ///   - decoder: A decoder that accepts the column elements' type.
    /// - Returns: A new column of decoded values.
    /// - Throws: `ColumnDecodingError` if an element fails to decode.
    public func decoded T  Decoder  _        T
         Decoder  throws    AnyColumn where T   Decodable
Decoder   TopLevelDecoder

    /// Decodes the data in each element of the column.
    ///
    /// - Parameters:
    ///   - type: The type of the value to decode.
    ///   - decoder: A decoder that accepts the column elements' type.
    /// - Throws: `ColumnDecodingError` if an element fails to decode.
    public mutating func decode T  Decoder  _        T
                Decoder  throws where T   Decodable  Decoder
TopLevelDecoder


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension AnyColumn

    /// Generates a column by encoding each element's value.
    ///
```

```
    /// - Parameters:
    ///   - type: The column underlying type.
    ///   - encoder: An encoder.
    ///
    /// - Returns: A new column of encoded values.
    ///
    /// - Throws: `ColumnEncodingError` when the encoder fails to
encode an element.
    public func encoded T  Encoder  _       T
        Encoder  throws    AnyColumn where T   Encodable
Encoder   TopLevelEncoder

    /// Encodes each element of the column.
    ///
    /// - Parameters:
    ///   - type: The type of elements in the column.
    ///   - encoder: An encoder.
    /// - Throws: `ColumnEncodingError` if an element fails to encode.
    public mutating func encode T  Encoder  _       T
             Encoder  throws where T   Encodable  Encoder
TopLevelEncoder


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension AnyColumn   RandomAccessCollection
MutableCollection

    /// The index of the initial element in the column.
    public var startIndex  Int   get

    /// The index of the final element in the column.
    public var endIndex  Int   get

    /// Returns the index immediately after an element index.
    /// - Parameter i: A valid index to an element in the column.
    public func index          Int     Int

    /// Returns the index immediately before an element index.
    /// - Parameter i: A valid index to an element in the column.
    public func index           Int     Int

    /// Accesses an element at an index.
    ///
    /// - Parameter position: A valid index in the column.
    public subscript           Int     Any

    /// Accesses a contiguous subrange of the elements.
    ///
    /// - Parameter range: A range of valid indices in the column.
```

```
    public subscript        Range Int      AnyColumnSlice

    /// Returns a slice of the column by selecting elements with a collection of
Booleans.
    ///
    /// - Parameter mask: A collection of Booleans.
    /// The method selects the column's elements that correspond to the `true`
elements in the collection.
    public subscript C        C     AnyColumnSlice where C
Collection  C Element    Bool   get

    /// Returns a Boolean that indicates whether the columns are equal.
    /// - Parameters:
    ///   - lhs: A type-erased column.
    ///   - rhs: Another type-erased column.
    public static func          AnyColumn       AnyColumn
Bool

    /// Hashes the essential components of the column by feeding them into a
hasher.
    /// - Parameter hasher: A hasher the method uses to combine the
components of the column.
    public func hash              inout Hasher

    /// A type representing the sequence's elements.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias Element   Any

    /// A type that represents a position in the collection.
    ///
    /// Valid indices consist of the position of every element and a
    /// "past the end" position that's not valid for use as a subscript
    /// argument.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias Index   Int

    /// A type that represents the indices that are valid for subscripting the
    /// collection, in ascending order.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias Indices   Range Int

    /// A type that provides the collection's iteration interface and
    /// encapsulates its iteration state.
    ///
    /// By default, a collection conforms to the `Sequence` protocol by
    /// supplying `IndexingIterator` as its associated `Iterator`
    /// type.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias Iterator   IndexingIterator AnyColumn
```

```
    /// A collection representing a contiguous subrange of this collection's
    /// elements. The subsequence shares indices with the original collection.
    ///
    /// The default subsequence type for collections that don't define their own
    /// is `Slice`.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias SubSequence   AnyColumnSlice


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension AnyColumn   CustomStringConvertible
CustomDebugStringConvertible  CustomReflectable

    /// A text representation of the column.
    public var description  String   get

    /// A text representation of the column suitable for debugging.
    public var debugDescription  String   get

    /// A mirror that reflects the column.
    public var customMirror  Mirror   get


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension AnyColumn

    /// Generates a column slice that contains unique elements.
    ///
    /// The method only adds the first of multiple elements with the same value
    /// --- the element with the smallest index ---
    /// to the slice.
    ///
    /// - Returns: A type-erased column slice.
    public func distinct      AnyColumnSlice


/// A type that represents a type-erased column.
///
/// `AnyColumnProtocol` defines the common functionality for type-erased
column types.
/// Its typed counterpart is ``ColumnProtocol``.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public protocol AnyColumnProtocol

    /// The name of the column type.
    var name  String   get set

    /// The number of elements in the column type.
    var count  Int   get
```

```
    /// The underlying type of the column type's elements.
    var wrappedElementType  any Any        get

    /// Retrieves an element at a position in the column type.
    ///
    /// - Parameter position: A valid index in the column type.
    subscript           Int     Any    get

    /// Retrieves a contiguous subrange of the column type's elements.
    ///
    /// - Parameter range: An integer range of valid indices in the column.
    subscript        Range Int      AnyColumnSlice   get


/// A prototype that creates type-erased columns.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public protocol AnyColumnPrototype

    /// The name of the column.
    var name  String   get set

    /// Creates a type-erased column.
    ///
    /// - Parameter capacity: The capacity of the new column.
    func makeColumn           Int     AnyColumn


/// A type-erased column slice.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public struct AnyColumnSlice   AnyColumnProtocol  Hashable

    /// The name of the slice's parent column.
    public var name  String

    /// The underlying type of the column's elements.
    public var wrappedElementType  any Any        get

    /// The number of elements in the column slice.
    public var count  Int   get

    /// The number of missing elements in the column slice.
    @available macOS 12  iOS 15  tvOS 15  watchOS 8
    public var missingCount  Int   get

    /// Returns a slice of the underlying typed column.
    ///
    /// When using this method, you must provide the correct underlying type.
    ///
```

```
    /// - Parameter type: The type of the slice's underlying parent column.
    /// - Returns: A typed column slice.
    public func assumingType T  _       T
DiscontiguousColumnSlice T

    /// Returns a Boolean that indicates whether the element at the index is
missing.
    ///
    /// - Parameter index: An index.
    public func isNil           Int     Bool

    /// The hash value.
    ///
    /// Hash values are not guaranteed to be equal across different executions of
    /// your program. Do not save hash values to use during a future execution.
    ///
    /// - Important: `hashValue` is deprecated as a `Hashable`
requirement. To
    ///     conform to `Hashable`, implement the `hash(into:)` requirement
instead.
    ///     The compiler provides an implementation for `hashValue` for you.
    public var hashValue  Int   get


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension AnyColumnSlice

    /// Generates a categorical summary of the column slice's elements.
    ///
    /// The method tries to cast the the untyped column slice to a typed column
slice before summarizing.
    /// Generating a summary for a typed column is faster and more efficient
than for an untyped column.
    public func summary      AnyCategoricalSummary


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension AnyColumnSlice   RandomAccessCollection
MutableCollection

    /// The index of the initial element in the column slice.
    public var startIndex  Int   get

    /// The index of the final element in the column slice.
    public var endIndex  Int   get

    /// Returns the index immediately after an element index.
    /// - Parameter i: A valid index to an element in the column slice.
    public func index           Int     Int
```

```
    /// Returns the index immediately before an element index.
    /// - Parameter i: A valid index to an element in the column slice.
    public func index            Int     Int

    /// Accesses an element at an index.
    ///
    /// - Parameter position: A valid index to an element in the column
slice.
    public subscript           Int     Any

    /// Accesses a contiguous range of elements.
    ///
    /// - Parameter range: A range of valid indices in the column slice.
    public subscript         Range Int      AnyColumnSlice

    /// Returns a Boolean that indicates whether the column slices are equal.
    /// - Parameters:
    ///   - lhs: A type-erased column slice.
    ///   - rhs: Another type-erased column slice.
    public static func           AnyColumnSlice
AnyColumnSlice     Bool

    /// Hashes the essential components of the column slice by feeding them
into a hasher.
    /// - Parameter hasher: A hasher the method uses to combine the
components of the column slice.
    public func hash              inout Hasher

    /// A type representing the sequence's elements.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias Element   Any

    /// A type that represents a position in the collection.
    ///
    /// Valid indices consist of the position of every element and a
    /// "past the end" position that's not valid for use as a subscript
    /// argument.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias Index   Int

    /// A type that represents the indices that are valid for subscripting the
    /// collection, in ascending order.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias Indices   Range Int

    /// A type that provides the collection's iteration interface and
    /// encapsulates its iteration state.
    ///
    /// By default, a collection conforms to the `Sequence` protocol by
    /// supplying `IndexingIterator` as its associated `Iterator`
```

```
    /// type.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias Iterator
IndexingIterator AnyColumnSlice

    /// A collection representing a contiguous subrange of this collection's
    /// elements. The subsequence shares indices with the original collection.
    ///
    /// The default subsequence type for collections that don't define their own
    /// is `Slice`.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias SubSequence   AnyColumnSlice


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension AnyColumnSlice   CustomStringConvertible
CustomDebugStringConvertible  CustomReflectable

    /// A text representation of the column slice.
    public var description  String   get

    /// A text representation of the column slice suitable for debugging.
    public var debugDescription  String   get

    /// A mirror that reflects the column slice.
    public var customMirror  Mirror   get


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension AnyColumnSlice

    /// Generates a column slice that contains unique elements.
    ///
    /// The method only adds the first of multiple elements with the same value
    /// --- the element with the smallest index ---
    /// to the slice.
    ///
    /// - Returns: A type-erased column slice.
    public func distinct      AnyColumnSlice


/// A CSV reading error.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public enum CSVReadingError   Error

    /// An error that indicates CSV data contains an invalid UTF-8 byte
sequence.
    ///
    /// - Parameters:
```

```
    ///    - row: The index of the row that contains the invalid sequence.
    ///    - column: The index of the column that contains the invalid
sequence.
    ///    - cellContents: The data that contains the invalid sequence.
    case badEncoding      Int           Int
Data

    /// An error that indicates the CSV reader doesn't support an encoding.
    ///
    /// The associated value contains a description of the error.
    case unsupportedEncoding String

    /// An error that indicates the CSV data contains a misplaced quote.
    ///
    /// - Parameters:
    ///    - row: The index of the row that contains the misplaced quote.
    ///    - column: The index of the column that contains the misplaced
quote.
    case misplacedQuote      Int           Int

    /// An error that indicates the CSV data contains a row with a mismatched
number of columns.
    ///
    /// - Parameters:
    ///    - row: The index of the row that contains the mismatched number of
columns.
    ///    - columns: The number of columns in the row.
    ///    - expected: The number of columns in the other rows.
    case wrongNumberOfColumns      Int           Int
          Int

    /// An error that indicates the CSV reader can't parse data in the file.
    ///
    /// - Parameters:
    ///    - row: The index of the row that contains the invalid data.
    ///    - column: The index of the column that contains the invalid data.
    ///    - type: The type the CSV reader expects.
    ///    - cellContents: The data the CSV reader can't parse.
    case failedToParse      Int           Int        CSVType
              Data

    /// An error that indicates the CSV is missing a required column.
    ///
    /// - Parameters:
    ///    - columnName: The name of the missing column.
    case missingColumn              String

    /// An error that indicates that the read operation requested rows beyond the
end of the CSV file.
    ///
```

```
    /// - Parameters:
    ///   - requested: The requested start row index.
    ///   - actual: The actual number of rows in the file.
    @available macOS 13  iOS 16  tvOS 16  watchOS 9
    case outOfBounds            Int          Int

    /// The index of the row that contains the error.
    public var row  Int   get

    /// The index of the column that contains the error.
    public var column  Int    get


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension CSVReadingError   CustomStringConvertible

    /// A text representation of the reading error.
    public var description  String   get


@available macOS 14  iOS 17  tvOS 17  watchOS 10
extension CSVReadingError   LocalizedError

    /// A localized message describing what error occurred.
    public var errorDescription  String    get


/// A set of CSV file-reading options.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public struct CSVReadingOptions

    /// A Boolean value that indicates whether the CSV file has a header row.
    ///
    /// Defaults to `true`.
    public var hasHeaderRow  Bool

    /// The set of strings that stores acceptable spellings for empty values.
    ///
    /// Defaults to `["", "#N/A", "#N/A N/A", "#NA", "N/A", "NA",
"NULL", "n/a", "null"]`.
    public var nilEncodings  Set String

    /// The set of strings that stores acceptable spellings for true Boolean values.
    ///
    /// Defaults to `["1", "True", "TRUE", "true"]`.
    public var trueEncodings  Set String

    /// The set of strings that stores acceptable spellings for false Boolean
values.
```

```
    ///
    /// Defaults to `["0", "False", "FALSE", "false"]`.
    public var falseEncodings  Set String

    /// The type to use for floating-point numeric values.
    ///
    /// Defaults to ``CSVType/double``.
    public var floatingPointType  CSVType

    /// An array of closures that parse a date from a string.
    public var dateParsers    String     Date

    /// A Boolean value that indicates whether to ignore empty lines.
    ///
    /// Defaults to `true`.
    public var ignoresEmptyLines  Bool

    /// A Boolean value that indicates whether to enable quoting.
    ///
    /// When `true`, the contents of a quoted field can contain special characters, such as the field
    /// delimiter and newlines. Defaults to `true`.
    public var usesQuoting  Bool

    /// A Boolean value that indicates whether to enable escaping.
    ///
    /// When `true`, you can escape special characters, such as the field delimiter, by prefixing them with
    /// the escape character, which is the backslash (`\`) by default. Defaults to `false`.
    public var usesEscaping  Bool

    /// The character that separates data fields in a CSV file, typically a comma.
    ///
    /// Defaults to comma (`,`).
    public var delimiter  Character   get

    /// The character that precedes other characters, such as quotation marks,
    /// so that the parser interprets them as literal characters instead of special ones.
    ///
    /// Defaults to backslash(`\`).
    public var escapeCharacter  Character   get

    /// Creates a set of options for reading a CSV file.
    ///
    /// - Parameters:
    ///   - hasHeaderRow: A Boolean value that indicates whether the CSV file has a header row. Defaults to `true`.
    ///   - nilEncodings: A list of recognized encodings of `nil`. Defaults
```

```
to
    ///     `["", "#N/A", "#N/A N/A", "#NA", "N/A", "NA",
"NULL", "n/a", "null"]`.
    ///   - trueEncodings: A list of acceptable encodings of `true`.
Defaults to `["1", "True", "TRUE", "true"]`.
    ///   - falseEncodings: A list of acceptable encodings of `false`.
Defaults to `["0", "False", "FALSE", "false"]`.
    ///   - floatingPointType: A type to use for floating-point numeric
values
    ///     (either ``CSVType/double`` or ``CSVType/float``).
    ///     Defaults to ``CSVType/double``.
    ///   - ignoresEmptyLines: A Boolean value that indicates whether to
ignore empty lines. Defaults to `true.`
    ///   - usesQuoting: A Boolean value that indicates whether the CSV
file uses quoting. Defaults to `true`.
    ///   - usesEscaping: A Boolean value that indicates whether the CSV
file uses escaping sequences. Defaults to
    ///     `false`.
    ///   - delimiter: A field delimiter. Defaults to comma (`,`).
    ///   - escapeCharacter: An escape character to use if
``usesEscaping`` is true. Defaults to backslash (`\`).
    public init                Bool   true
Set String     ""  "#N/A"  "#N/A N/A"  "#NA"  "N/A"  "NA"
"NULL"  "n/a"  "nil"  "null"                  Set String
 "1"  "True"  "TRUE"  "true"                   Set String
 "0"  "False"  "FALSE"  "false"                      CSVType
                          Bool   true               Bool
true               Bool   false             Character
          ","                  Character             "\\"

    /// Adds a date parsing strategy.
    /// - Parameter strategy: A parsing strategy that has a string input
and a date output.
    public mutating func addDateParseStrategy T   _
T  where T   ParseStrategy  T ParseInput    String
T ParseOutput    Date


/// Represents the value types in a CSV file.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public enum CSVType   Sendable

    /// An integer type.
    case integer

    /// A Boolean type.
    case boolean

    /// A single-precision floating-point type.
    case float
```

```
        /// A double-precision floating-point type.
        case double

        /// A date type.
        case date

        /// A string type.
        case string

        /// A binary data type.
        case data

        /// Returns a Boolean value indicating whether two values are equal.
        ///
        /// Equality is the inverse of inequality. For any values `a` and `b`,
        /// `a == b` implies that `a != b` is `false`.
        ///
        /// - Parameters:
        ///   - lhs: A value to compare.
        ///   - rhs: Another value to compare.
        public static func        CSVType     CSVType     Bool

        /// Hashes the essential components of this value by feeding them into the
        /// given hasher.
        ///
        /// Implement this method to conform to the `Hashable` protocol. The
        /// components used for hashing must be the same as the components
compared
        /// in your type's `==` operator implementation. Call
`hasher.combine(_:)`
        /// with each of these components.
        ///
        /// - Important: In your implementation of `hash(into:)`,
        ///   don't call `finalize()` on the `hasher` instance provided,
        ///   or replace it with a different instance.
        ///   Doing so may become a compile-time error in the future.
        ///
        /// - Parameter hasher: The hasher to use when combining the
components
        ///   of this instance.
        public func hash                inout Hasher

        /// The hash value.
        ///
        /// Hash values are not guaranteed to be equal across different executions of
        /// your program. Do not save hash values to use during a future execution.
        ///
        /// - Important: `hashValue` is deprecated as a `Hashable`
requirement. To
```

```
    ///     conform to `Hashable`, implement the `hash(into:)` requirement instead.
    ///     The compiler provides an implementation for `hashValue` for you.
    public var hashValue  Int    get


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension CSVType   Equatable


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension CSVType   Hashable


/// A CSV writing error.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public enum CSVWritingError   Error

    /// An error that indicates CSV data contains an invalid UTF-8 byte sequence.
    ///
    /// - Parameters:
    ///   - row: The index of the row that contains the invalid sequence.
    ///   - column: The index of the column that contains the invalid sequence.
    ///   - data: The data that contains the invalid sequence.
    case badEncoding      Int          String  Data

    /// The index of the row that contains the error.
    public var row  Int    get

    /// The index of the column that contains the error.
    public var column  String    get


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension CSVWritingError   CustomStringConvertible

    /// A text representation of the writing error.
    public var description  String    get


/// A set of CSV file-writing options.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public struct CSVWritingOptions

    /// A Boolean value that indicates whether to write a header with the column names.
    ///
    /// Defaults to `true`.
```

```
    public var includesHeader: Bool

    /// The format the CSV file generator uses to create date strings.
    ///
    /// Defaults to `nil`, which uses ISO 8601 encoding.
    @available(*, deprecated, message: "Use dateFormatter instead.")
    public var dateFormat: String?

    /// A closure that maps dates to their string representations.
    ///
    /// Defaults to ISO 8601 encoding.
    @available(macOS 12.3, iOS 15.4, tvOS 15.4, watchOS 8.5, *)
    public var dateFormatter: ((Date) -> String)?

    /// The string the CSV file generator uses to represent nil values.
    ///
    /// Defaults to an empty string.
    public var nilEncoding: String

    /// The string the CSV file generator uses to represent true Boolean values.
    ///
    /// Defaults to `true`.
    public var trueEncoding: String

    /// The string the CSV file generator uses to represent false Boolean values.
    ///
    /// Defaults to `false`.
    public var falseEncoding: String

    /// The string the CSV file generator uses to represent a newline sequence.
    ///
    /// Defaults to a line feed.
    public var newline: String

    /// The character the CSV file generator uses to separate data fields in a CSV file.
    ///
    /// Defaults to comma (`,`).
    public var delimiter: Character

    /// Creates the default set of options for writing a CSV file.
    public init

    /// Creates a set of options for writing a CSV file.
    /// - Parameters:
    ///   - includesHeader: A Boolean value that indicates whether to write a header with the column names. Defaults to
    ///     `true`.
```

```
    ///   - dateFormatter: The format the CSV file generator uses to
create date strings.
    ///   - nilEncoding: The spelling for nil values. Defaults to an empty
string.
    ///   - trueEncoding: The spelling for true Boolean values. Defaults to
`true`.
    ///   - falseEncoding: The spelling for false Boolean values. Defaults
to `false`.
    ///   - newline: The newline sequence. Defaults to a line feed.
    ///   - delimiter: The field delimiter. Defaults to comma (`,`).
    @available                          "Use dateFormatter
instead or dateFormat."
    public init                 Bool   true
String                String   ""                String
"true"                 String   "false"           String
"\n"             Character   ","

    /// Creates a set of options for writing a CSV file.
    /// - Parameters:
    ///   - includesHeader: A Boolean value that indicates whether to write
a header with the column names. Defaults to
    ///     `true`.
    ///   - nilEncoding: The spelling for nil values. Defaults to an empty
string.
    ///   - trueEncoding: The spelling for true Boolean values. Defaults to
`true`.
    ///   - falseEncoding: The spelling for false Boolean values. Defaults
to `false`.
    ///   - newline: The newline sequence. Defaults to a line feed.
    ///   - delimiter: The field delimiter. Defaults to comma (`,`).
    public init                 Bool   true
String   ""                String   "true"
String   "false"           String   "\n"            Character
  ","


/// A categorical summary of a collection's elements.
///
/// Each categorical summary has 5 statistics about a collection:
///   - `someCount`: The number of non-missing elements.
///   - `noneCount` The number of missing elements.
///   - `uniqueCount`: The number of unique elements.
///   - `totalCount`: The total number of elements.
///   - `mode`: An array of the most common values.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public struct CategoricalSummary Element    Hashable
CustomDebugStringConvertible where Element   Hashable

    /// The number of non-missing elements in the column.
    public var someCount  Int
```

```
        /// The number of missing elements in the column.
        public var noneCount  Int

        /// The total number of elements in the column.
        public var totalCount  Int   get

        /// The number of elements with distinct values in a column that excludes
missing elements.
        public var uniqueCount  Int

        /// The most common values in a column, ignoring missing elements.
        ///
        /// The summary orders the elements based on their original locations within
the column.
        public var mode   Element

        /// Creates a categorical summary with default values.
        public init

        /// Creates a categorical summary.
        ///
        /// - Parameters:
        ///   - someCount: The number of elements in a column, excluding
missing elements.
        ///   - noneCount: The number of missing elements in the column.
        ///   - uniqueCount: The number of elements with distinct values in a
column, ignoring missing elements.
        ///   - mode: The most common values in a column, ignoring missing
elements.
        public init            Int             Int
Int         Element

        /// A text representation of the summary's statistics suitable for debugging.
        public var debugDescription  String   get

        /// Hashes the essential components of this value by feeding them into the
        /// given hasher.
        ///
        /// Implement this method to conform to the `Hashable` protocol. The
        /// components used for hashing must be the same as the components
compared
        /// in your type's `==` operator implementation. Call
`hasher.combine(_:)`
        /// with each of these components.
        ///
        /// - Important: In your implementation of `hash(into:)`,
        ///   don't call `finalize()` on the `hasher` instance provided,
        ///   or replace it with a different instance.
        ///   Doing so may become a compile-time error in the future.
```

```
    ///
    /// - Parameter hasher: The hasher to use when combining the components
    ///   of this instance.
    public func hash                inout Hasher

    /// Returns a Boolean value indicating whether two values are equal.
    ///
    /// Equality is the inverse of inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is `false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to compare.
    public static func        CategoricalSummary Element CategoricalSummary Element      Bool

    /// The hash value.
    ///
    /// Hash values are not guaranteed to be equal across different executions of
    /// your program. Do not save hash values to use during a future execution.
    ///
    /// - Important: `hashValue` is deprecated as a `Hashable` requirement. To
    ///   conform to `Hashable`, implement the `hash(into:)` requirement instead.
    ///   The compiler provides an implementation for `hashValue` for you.
    public var hashValue  Int   get


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension CategoricalSummary   Sendable where Element Sendable


/// A column in a data frame.
///
/// A column is a <doc://com.apple.documentation/documentation/Swift/Collection> that contains
/// values of a specific type, including:
/// - <doc://com.apple.documentation/documentation/Swift/Int>
/// - <doc://com.apple.documentation/documentation/Swift/Double>
/// - <doc://com.apple.documentation/documentation/Swift/String>
///
/// Each element in a column is an
/// <doc://com.apple.documentation/documentation/Swift/Optional>
/// of the column's type. Each `nil` element represents a missing value.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public struct Column WrappedElement    OptionalColumnProtocol
```

```
    /// The type of the column's elements, which is an optional type of the column's type.
    public typealias Element = WrappedElement

    /// The name of the column.
    public var name: String

    /// The number of elements in the column.
    public var count: Int { get }

    /// The number of missing elements in the column.
    @available(macOS 12, iOS 15, tvOS 15, watchOS 8)
    public var missingCount: Int { get }

    /// The underlying type of the column’s elements.
    public var wrappedElementType: any Any.Type { get }

    /// A prototype that creates type-erased columns with the same underlying type as the column slice.
    ///
    /// Use a type-erased column prototype to create new columns of the same type as the slice's parent column
    /// without explicitly knowing what type it is.
    public var prototype: any AnyColumnPrototype { get }

    /// Creates a column with a name and a capacity.
    ///
    /// - Parameters:
    ///   - name: A column name.
    ///   - capacity: An integer that represents the number of elements the column can initially store.
    public init(name: String, capacity: Int)

    /// Creates a column with a column identifier and a capacity.
    ///
    /// - Parameters:
    ///   - id: A column identifier.
    ///   - capacity: An integer that represents the number of elements the column can initially store.
    public init(_ id: ColumnID<WrappedElement>, capacity: Int)

    /// Creates a column with a name and a sequence of optional values.
    ///
    /// - Parameters:
    ///   - name: A column name.
    ///   - contents: A sequence of optional elements.
    public init<S>(name: String, contents: S) where S: Sequence, S.Element == WrappedElement
```

```
    /// Creates a column with a name and a sequence of nonoptional values.
    ///
    /// - Parameters:
    ///   - name: A column name.
    ///   - contents: A sequence of nonoptional elements.
    public init S        String            S  where
WrappedElement    S Element  S   Sequence

    /// Creates a column with a column identifier and a sequence of optional
values.
    ///
    /// - Parameters:
    ///   - id: A column identifier.
    ///   - contents: A sequence of optional elements.
    public init S  _     ColumnID S Element             S
where S   Sequence  S Element    WrappedElement

    /// Creates a column with an identifier and a sequence of nonoptional values.
    ///
    /// - Parameters:
    ///   - id: A column identifier.
    ///   - contents: A sequence of elements.
    public init S  _     ColumnID S Element             S
where WrappedElement    S Element  S   Sequence

    /// Creates a column from a column slice.
    ///
    /// - Parameter slice: A column slice.
    public init _        ColumnSlice WrappedElement

    /// Appends an optional value to the column.
    ///
    /// - Parameter element: A nonoptional element.
    public mutating func append _
Column WrappedElement  Element

    /// Appends a nonoptional value to the column.
    ///
    /// - Parameter element: An optional element.
    public mutating func append _          WrappedElement

    /// Appends a sequence of optional values to the column.
    ///
    /// - Parameter sequence: A sequence of optional elements.
    public mutating func append S                        S
where S   Sequence  S Element    WrappedElement

    /// Appends a sequence of nonoptional values to the column.
    ///
```

```
    /// - Parameter sequence: A sequence of nonoptional elements.
    public mutating func append S                          S
where WrappedElement    S Element  S   Sequence

    /// Removes an element from the column.
    ///
    /// - Parameter index: The element's location in the column.
    public mutating func remove           Int

    /// Creates a new column by applying a transformation to every element.
    ///
    /// - Parameter transform: A transformation closure.
    public func map T  _
 Column WrappedElement  Element  throws    T   rethrows
Column T

    /// Creates a new column by applying the transformation to every element
that isn't missing.
    ///
    /// - Parameter transform: A transformation closure.
    public func mapNonNil T  _              WrappedElement
throws    T   rethrows    Column T

    /// Applies a transformation to every element in the column.
    ///
    /// - Parameter transform: A transformation closure.
    public mutating func transform _
 Column WrappedElement  Element  throws
Column WrappedElement  Element  rethrows

    /// Applies a transformation to every element that isn't missing.
    ///
    /// - Parameter transform: A transformation closure.
    public mutating func transform _
 WrappedElement  throws    WrappedElement  rethrows

    /// Generates a slice that contains the elements that satisfy a predicate.
    ///
    /// - Parameter isIncluded: A predicate closure that returns a
Boolean.
    /// The method uses the closure to determine whether it includes an element
in the slice.
    public func filter _
 Column WrappedElement  Element  throws    Bool  rethrows
DiscontiguousColumnSlice WrappedElement

    /// Generates a type-erased copy of the column.
    public func eraseToAnyColumn      AnyColumn

    /// A type that represents the indices that are valid for subscripting the
```

```
    /// collection, in ascending order.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias Indices
Range Column WrappedElement  Index

    /// A type that provides the collection's iteration interface and
    /// encapsulates its iteration state.
    ///
    /// By default, a collection conforms to the `Sequence` protocol by
    /// supplying `IndexingIterator` as its associated `Iterator`
    /// type.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias Iterator
IndexingIterator Column WrappedElement


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension Column where WrappedElement   Hashable

    /// Generates a categorical summary of the column's elements.
    public func summary
CategoricalSummary WrappedElement


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension Column   RandomAccessCollection  MutableCollection

    /// The index of the initial element in the column.
    public var startIndex  Int   get

    /// The index of the final element in the column.
    public var endIndex  Int   get

    /// Returns the index immediately after an element index.
    /// - Parameter i: A valid index to an element in the column.
    public func index          Int     Int

    /// Returns the index immediately before an element index.
    /// - Parameter i: A valid index to an element in the column.
    public func index           Int     Int

    /// Accesses an element at an index.
    ///
    /// - Parameter position: A valid index to an element in the column.
    public subscript           Int
Column WrappedElement  Element

    /// Accesses a contiguous range of elements.
    ///
```

```
    /// - Parameter bounds: A range of valid indices in the column.
    public subscript          Range Int
ColumnSlice WrappedElement

    /// Accesses a contiguous range of elements with a range expression.
    ///
    /// - Parameter range: An integer range expression that represents
valid indices in the column.
    @inlinable public subscript R          R
ColumnSlice WrappedElement  where R   RangeExpression  R Bound
   Int

    /// Returns a column slice that includes elements that correspond to a
collection of Booleans.
    ///
    /// - Parameter mask: A Boolean collection. The subscript returns a
slice that includes the column elements
    /// that correspond to the `true` elements in `mask`.
    ///
    /// You can create a Boolean column for this subscript by comparing a
column to a value
    /// of the column elements' type.
    ///
    /// ```swift
    /// let followerColumn = artists["Followers",
Int.self].filled(with: 0)
    /// let popularArtists = artists[followerColumn >
10_000_000]
    /// ```
    public subscript C         C
DiscontiguousColumnSlice WrappedElement  where C   Collection
C Element    Bool   get

    /// This method always returns `nil` without calling `body`.
    ///
    /// Use the version of this method that uses a buffer of non-optional
elements.
    public func withContiguousStorageIfAvailable R  _
 UnsafeBufferPointer WrappedElement    throws    R  rethrows
   R

    /// This method always returns `nil` without calling `body`.
    ///
    /// Use the version of this method that uses a buffer of non-optional
elements.
    public mutating func
withContiguousMutableStorageIfAvailable R  _        inout
UnsafeMutableBufferPointer WrappedElement    throws    R
rethrows    R
```

```
    /// Call `body(buffer)`, where `buffer` provides access to the non-optional contiguous storage of the entire
    /// column. If the column contains missing values, `body` is not called and `nil` is returned.
    ///
    /// The optimizer can often eliminate bounds- and uniqueness-checking within an algorithm. When that fails,
    /// however, invoking the same algorithm on `body`'s argument may let you trade safety for speed.
    ///
    /// - Parameters:
    ///   - body: a closure to be executed using the elements of this collection.
    ///   - buffer: a buffer to the contiguous storage of this collection.
    /// - Returns: the value returned by `body`, or `nil`.
    @available macOS 13  iOS 16  tvOS 16  watchOS 9
    public func withContiguousStorageIfAvailable R  _
 UnsafeBufferPointer WrappedElement   throws    R  rethrows
R

    /// Call `body(buffer)`, where `buffer` provides access to the non-optional contiguous mutable storage of the
    /// entire column. If the column contains missing values, `body` is not called and `nil` is returned.
    ///
    /// The optimizer can often eliminate bounds- and uniqueness-checking within an algorithm. When that fails,
    /// however, invoking the same algorithm on `body`'s argument may let you trade safety for speed.
    ///
    /// - Note: `buffer` must not be replaced by `body`.
    ///
    /// - Parameters:
    ///   - body: a closure to be executed using the elements of this collection.
    ///   - buffer: a buffer to the mutable contiguous storage of this collection.
    /// - Returns: the value returned by `body`, or `nil`.
    @available macOS 13  iOS 16  tvOS 16  watchOS 9
    public mutating func
withContiguousMutableStorageIfAvailable R  _        inout
UnsafeMutableBufferPointer WrappedElement   throws    R
rethrows    R

    /// A type that represents a position in the collection.
    ///
    /// Valid indices consist of the position of every element and a
    /// "past the end" position that's not valid for use as a subscript
    /// argument.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias Index   Int
```

```
    /// A collection representing a contiguous subrange of this collection's
    /// elements. The subsequence shares indices with the original collection.
    ///
    /// The default subsequence type for collections that don't define their own
    /// is `Slice`.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias SubSequence   ColumnSlice WrappedElement


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension Column   Equatable where WrappedElement   Equatable


    /// Returns a Boolean value indicating whether two values are equal.
    ///
    /// Equality is the inverse of inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is `false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to compare.
    public static func        Column WrappedElement
Column WrappedElement      Bool


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension Column   Hashable where WrappedElement   Hashable


    /// Generates a discontiguous slice that contains unique elements.
    ///
    /// The method only adds the first of multiple elements with the same value
    /// --- the element with the smallest index ---
    /// to the slice.
    ///
    /// - Returns: A discontiguous column slice.
    public func distinct
DiscontiguousColumnSlice WrappedElement


    /// Hashes the essential components of this value by feeding them into the
    /// given hasher.
    ///
    /// Implement this method to conform to the `Hashable` protocol. The
    /// components used for hashing must be the same as the components
compared
    /// in your type's `==` operator implementation. Call
`hasher.combine(_:)`
    /// with each of these components.
    ///
```

```
    /// - Important: In your implementation of `hash(into:)`,
    ///   don't call `finalize()` on the `hasher` instance provided,
    ///   or replace it with a different instance.
    ///   Doing so may become a compile-time error in the future.
    ///
    /// - Parameter hasher: The hasher to use when combining the components
    ///   of this instance.
    public func hash                inout Hasher

    /// The hash value.
    ///
    /// Hash values are not guaranteed to be equal across different executions of
    /// your program. Do not save hash values to use during a future execution.
    ///
    /// - Important: `hashValue` is deprecated as a `Hashable` requirement. To
    ///   conform to `Hashable`, implement the `hash(into:)` requirement instead.
    ///   The compiler provides an implementation for `hashValue` for you.
    public var hashValue  Int   get


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension Column   Encodable where WrappedElement   Encodable


    /// Encodes this value into the given encoder.
    ///
    /// If the value fails to encode anything, `encoder` will encode an empty
    /// keyed container in its place.
    ///
    /// This function throws an error if any values are invalid for the given
    /// encoder's format.
    ///
    /// - Parameter encoder: The encoder to write data to.
    public func encode              any Encoder  throws


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension Column   Decodable where WrappedElement   Decodable


    /// Creates a new instance by decoding from the given decoder.
    ///
    /// This initializer throws an error if reading from the decoder fails, or
    /// if the data read is corrupted or otherwise invalid.
    ///
    /// - Parameter decoder: The decoder to read data from.
    public init                 any Decoder  throws
```

```
@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension Column   Sendable where WrappedElement   Sendable


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension Column

    /// Generates a column by decoding each element's data.
    ///
    /// - Parameters:
    ///   - type: The decodable value's type.
    ///   - decoder: A decoder.
    ///
    /// - Returns: A new column of decoded values.
    ///
    /// - Throws: `ColumnDecodingError` when the decoder fails to
decode an element.
    public func decoded T  Decoder  _       T
        Decoder  throws    Column T  where WrappedElement
Decoder Input  T   Decodable  Decoder   TopLevelDecoder


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension Column

    /// Modifies a column by adding a value to each element.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A value of the same type as the column's elements.
    public static func          inout Column WrappedElement
     WrappedElement  where WrappedElement   AdditiveArithmetic

    /// Modifies a column by subtracting a value from each element.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A value of the same type as the column's elements.
    public static func          inout Column WrappedElement
     WrappedElement  where WrappedElement   AdditiveArithmetic

    /// Modifies a column by multiplying each element by a value.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A value of the same type as the column's elements.
    public static func          inout Column WrappedElement
```

```
 WrappedElement  where WrappedElement   Numeric

    /// Modifies an integer column by dividing each element by a value.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A value of the same type as the column's elements.
    public static func          inout Column WrappedElement
     WrappedElement  where WrappedElement   BinaryInteger

    /// Modifies a floating-point column by dividing each element by a value.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A value of the same type as the column's elements.
    public static func          inout Column WrappedElement
     WrappedElement  where WrappedElement   FloatingPoint


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension Column

    /// Modifies a column by adding each value in a collection to
    /// the corresponding element in the column.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the
column's elements.
    public static func     C       inout
Column WrappedElement        C  where WrappedElement
AdditiveArithmetic  WrappedElement    C Element  C
Collection

    /// Modifies a column by adding each optional value in a collection to
    /// the corresponding element in the column.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the
column's elements.
    public static func     C       inout
Column WrappedElement        C  where WrappedElement
AdditiveArithmetic  C   Collection  C Element
WrappedElement

    /// Modifies a column by subtracting each value in a collection from
    /// the corresponding element in the column.
    ///
    /// - Parameters:
```

```
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the column's elements.
    public static func += <C>(lhs: inout Column<WrappedElement>, rhs: C) where WrappedElement: AdditiveArithmetic, WrappedElement == C.Element, C: Collection

    /// Modifies a column by subtracting each optional value in a collection from
    /// the corresponding element in the column.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the column's elements.
    public static func -= <C>(lhs: inout Column<WrappedElement>, rhs: C) where WrappedElement: AdditiveArithmetic, C == Collection<C.Element>, C.Element == WrappedElement

    /// Modifies a column by multiplying each element in the column by
    /// the corresponding value in a collection.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the column's elements.
    public static func *= <C>(lhs: inout Column<WrappedElement>, rhs: C) where WrappedElement: Numeric, WrappedElement == C.Element, C: Collection

    /// Modifies a column by multiplying each element in the column by
    /// the corresponding optional value in a collection.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the column's elements.
    public static func *= <C>(lhs: inout Column<WrappedElement>, rhs: C) where WrappedElement: Numeric, C == Collection<C.Element>, C.Element == WrappedElement

    /// Modifies an integer column by dividing each element in the column by
    /// the corresponding value in a collection.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the column's elements.
    public static func /= <C>(lhs: inout
```

```
Column WrappedElement        C  where WrappedElement
BinaryInteger  WrappedElement    C Element  C   Collection

    /// Modifies an integer column by dividing each element in the column by
    /// the corresponding optional value in a collection.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the
column's elements.
    public static func     C       inout
Column WrappedElement        C  where WrappedElement
BinaryInteger  C   Collection  C Element    WrappedElement

    /// Modifies a floating-point column by dividing each element in the column
by
    /// the corresponding value in a collection.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the
column's elements.
    public static func     C       inout
Column WrappedElement        C  where WrappedElement
FloatingPoint  WrappedElement    C Element  C   Collection

    /// Modifies a floating-point column by dividing each element in the column
by
    /// the corresponding optional value in a collection.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the
column's elements.
    public static func     C       inout
Column WrappedElement        C  where WrappedElement
FloatingPoint  C   Collection  C Element    WrappedElement


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension Column where WrappedElement   Encodable

    /// Generates a column by encoding each element's value.
    ///
    /// - Parameters:
    ///   - encoder: An encoder.
    ///
    /// - Returns: A new column of encoded values.
    ///
    /// - Throws: `ColumnEncodingError` when the encoder fails to
```

```
encode an element.
    public func encoded Encoder                    Encoder
throws    Column Encoder Output  where Encoder
TopLevelEncoder


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension Column where WrappedElement   Comparable

    /// Returns the element with the lowest value, ignoring missing elements.
    public func min      Column WrappedElement  Element

    /// Returns the element with the highest value, ignoring missing elements.
    public func max      Column WrappedElement  Element

    /// Returns the index of the element with the lowest value, ignoring missing
elements.
    public func argmin      Int

    /// Returns the index of the element with the highest value, ignoring missing
elements.
    public func argmax      Int


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension Column where WrappedElement   AdditiveArithmetic

    /// Returns the sum of the column's elements, ignoring missing elements.
    public func sum      WrappedElement


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension Column where WrappedElement   FloatingPoint

    /// Returns the mean average of the floating-point column's elements,
ignoring missing elements.
    public func mean      Column WrappedElement  Element

    /// Returns the standard deviation of the floating-point column's elements,
ignoring missing elements.
    ///
    /// - Parameter deltaDegreesOfFreedom: A nonnegative integer.
    /// The method calculates the standard deviation's divisor by subtracting this
parameter from the number of
    /// non-`nil` elements (`n - deltaDegreesOfFreedom` where `n` is
the number of non-`nil` elements).
    ///
    /// - Returns: The standard deviation; otherwise, `nil` if there are
fewer than
    /// `deltaDegreesOfFreedom + 1` non-`nil` items in the column.
```

```
    public func standardDeviation                                Int
1     Column WrappedElement  Element


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension Column where WrappedElement   BinaryInteger

    /// Returns the mean average of the integer column's elements, ignoring
missing elements.
    public func mean      Double

    /// Returns the standard deviation of the integer column's elements, ignoring
missing elements.
    ///
    /// - Parameter deltaDegreesOfFreedom: A nonnegative integer.
    /// The method calculates the standard deviation's divisor by subtracting this
parameter from the number of
    /// non-`nil` elements (`n - deltaDegreesOfFreedom` where `n` is
the number of non-`nil` elements).
    ///
    /// - Returns: The standard deviation; otherwise, `nil` if there are
fewer than
    /// `deltaDegreesOfFreedom + 1` non-`nil` items in the column.
    public func standardDeviation                                Int
1     Double


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension Column   CustomStringConvertible
CustomDebugStringConvertible  CustomReflectable

    /// A text representation of the column.
    public var description  String   get

    /// A text representation of the column suitable for debugging.
    public var debugDescription  String   get

    /// A mirror that reflects the column.
    public var customMirror  Mirror   get


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension Column where WrappedElement   BinaryFloatingPoint

    /// Generates a numeric summary of the floating-point column's elements.
    public func numericSummary
NumericSummary WrappedElement


@available macOS 12  iOS 15  tvOS 15  watchOS 8
```

```swift
extension Column where WrappedElement   BinaryInteger

    /// Generates a numeric summary of the integer column's elements.
    public func numericSummary      NumericSummary Double


/// A column decoding error.
///
/// This error wraps a decoding error and includes the column name and row index where the decoding error occurs.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public struct ColumnDecodingError   Error  LocalizedError CustomDebugStringConvertible

    /// The name of the column with the error.
    public var columnName  String

    /// The index of the column's element with the error.
    public var rowIndex  Int

    /// The underlying decoding error.
    public var decodingError  DecodingError

    /// Creates a column decoding error.
    ///
    /// - Parameters:
    ///   - columnName: The name of the column with the error.
    ///   - rowIndex: The index of the column's element with the error.
    ///   - decodingError: An underlying decoding error.
    public init             String            Int
                DecodingError

    /// A text representation of the column decoding error suitable for debugging.
    public var debugDescription  String   get


/// A column encoding error.
///
/// An error bundles an
/// <doc://com.apple.documentation/documentation/Swift/EncodingError>
/// with the row and column that produces the error.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public struct ColumnEncodingError   Error  LocalizedError CustomDebugStringConvertible

    /// The name of the column with the error.
    public var columnName  String

    /// The index of the column's element with the error.
```

```
    public var rowIndex  Int

    /// The underlying encoding error.
    public var encodingError  EncodingError

    /// Creates a column encoding error.
    ///
    /// - Parameters:
    ///   - columnName: The name of the column with the error.
    ///   - rowIndex: The index of the column's element with the error.
    ///   - encodingError: An underlying encoding error.
    public init             String           Int
                EncodingError

    /// A text representation of the column encoding error suitable for debugging.
    public var debugDescription  String   get


/// A column identifier that stores a column's name and the type of its elements.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public struct ColumnID T    Sendable

    /// The name of the column the identifier represents.
    public var name  String

    /// Creates a column identifier.
    ///
    /// - Parameters:
    ///   - name: The name of a column.
    ///   - type: The type of the column's elements.
    public init _       String  _       T


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension ColumnID   CustomStringConvertible

    /// A text representation of the column identifier.
    public var description  String   get


/// A type that represents a column.
///
/// `ColumnProtocol` defines the common functionality for typed column types.
/// Its type-erased counterpart is ``AnyColumnProtocol``.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public protocol ColumnProtocol
BidirectionalCollection

    /// The name of the column.
```

```
    var name  String   get set


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension ColumnProtocol where Self Element
AdditiveArithmetic

    /// Generates a column by adding each element in a column type to the
corresponding elements of another.
    /// - Parameters:
    ///   - lhs: A column type.
    ///   - rhs: Another column type.
    /// - Returns: A new column.
    public static func         Self       Self
Column Self Element

    /// Generates a column by subtracting each element in a column type from
the corresponding elements of another.
    /// - Parameters:
    ///   - lhs: A column type.
    ///   - rhs: Another column type.
    /// - Returns: A new column.
    public static func         Self       Self
Column Self Element


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension ColumnProtocol where Self Element   Numeric

    /// Generates a column by multiplying each element in a column type by the
corresponding elements of another.
    /// - Parameters:
    ///   - lhs: A column type.
    ///   - rhs: Another column type.
    /// - Returns: A new column.
    public static func         Self       Self
Column Self Element


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension ColumnProtocol where Self Element   BinaryInteger

    /// Generates an integer column by dividing each element in a column type
by the corresponding elements of another.
    /// - Parameters:
    ///   - lhs: A column type.
    ///   - rhs: Another column type.
    /// - Returns: A new column.
    public static func         Self       Self
Column Self Element
```

```
@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension ColumnProtocol where Self Element   FloatingPoint

    /// Generates a floating-point column by dividing each element in a column
type
    /// by the corresponding elements of another.
    /// - Parameters:
    ///   - lhs: A column type.
    ///   - rhs: Another column type.
    /// - Returns: A new column.
    public static func         Self       Self
Column Self Element


extension ColumnProtocol

    /// Generates a column by adding a value to each element in a column.
    /// - Parameters:
    ///   - lhs: A column type.
    ///   - rhs: A value of the same type as the column.
    /// - Returns: A new column.
    @available macOS 12  iOS 15  tvOS 15  watchOS 8
    public static func         Self       Self Element
Column Self Element  where Self Element   AdditiveArithmetic

    /// Generates a column by adding each element in a column to a value.
    /// - Parameters:
    ///   - lhs: A value of the same type as the column.
    ///   - rhs: A column type.
    /// - Returns: A new column.
    @available macOS 12  iOS 15  tvOS 15  watchOS 8
    public static func         Self Element       Self
Column Self Element  where Self Element   AdditiveArithmetic

    /// Generates a column by subtracting a value from each element in a
column.
    /// - Parameters:
    ///   - lhs: A column type.
    ///   - rhs: A value of the same type as the column.
    /// - Returns: A new column.
    @available macOS 12  iOS 15  tvOS 15  watchOS 8
    public static func         Self       Self Element
Column Self Element  where Self Element   AdditiveArithmetic

    /// Generates a column by subtracting each element in a column from a
value.
    /// - Parameters:
    ///   - lhs: A value of the same type as the column.
```

```
    ///   - rhs: A column type.
    /// - Returns: A new column.
    @available macOS 12  iOS 15  tvOS 15  watchOS 8
    public static func         Self Element       Self
Column Self Element  where Self Element   AdditiveArithmetic


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension ColumnProtocol where Self Element   Numeric

    /// Generates a column by multiplying each element in a column by a value.
    /// - Parameters:
    ///   - lhs: A column type.
    ///   - rhs: A value of the same type as the column.
    /// - Returns: A new column.
    public static func         Self       Self Element
Column Self Element

    /// Generates a column by multiplying a value by each element in a column.
    /// - Parameters:
    ///   - lhs: A value of the same type as the column.
    ///   - rhs: A column type.
    /// - Returns: A new column.
    public static func         Self Element       Self
Column Self Element


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension ColumnProtocol where Self Element   BinaryInteger

    /// Generates an integer column by dividing each element in a column by a
value.
    /// - Parameters:
    ///   - lhs: A column type.
    ///   - rhs: A value of the same type as the column.
    /// - Returns: A new column.
    public static func         Self       Self Element
Column Self Element

    /// Generates an integer column by dividing a value by each element in a
column.
    /// - Parameters:
    ///   - lhs: A value of the same type as the column.
    ///   - rhs: A column type.
    /// - Returns: A new column.
    public static func         Self Element       Self
Column Self Element


@available macOS 12  iOS 15  tvOS 15  watchOS 8
```

```
extension ColumnProtocol where Self Element   FloatingPoint

    /// Generates a floating-point column by dividing each element in a column
by a value.
    /// - Parameters:
    ///   - lhs: A column type.
    ///   - rhs: A value of the same type as the column.
    /// - Returns: A new column.
    public static func         Self       Self Element
Column Self Element

    /// Generates a floating-point column by dividing a value by each element in
a column.
    /// - Parameters:
    ///   - lhs: A value of the same type as the column.
    ///   - rhs: A column type.
    /// - Returns: A new column.
    public static func         Self Element       Self
Column Self Element


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension ColumnProtocol where Self Element   Comparable

    /// Returns a Boolean array that indicates whether the corresponding
element of a column type
    /// is less than a value.
    ///   - lhs: A column type.
    ///   - rhs: A value of the same type as the column.
    /// - Returns: A Boolean array.
    public static func         Self       Self Element
 Bool

    /// Returns a Boolean array that indicates whether the value
    /// is less than the corresponding element of a column type.
    ///   - lhs: A value of the same type as the column.
    ///   - rhs: A column type.
    /// - Returns: A Boolean array.
    public static func         Self Element       Self
 Bool

    /// Returns a Boolean array that indicates whether the corresponding
element of a column type
    /// is less than or equal to a value.
    ///   - lhs: A column type.
    ///   - rhs: A value of the same type as the column.
    /// - Returns: A Boolean array.
    public static func          Self       Self Element
 Bool
```

```
    /// Returns a Boolean array that indicates whether the value
    /// is less than or equal to the corresponding element of a column type.
    ///   - lhs: A value of the same type as the column.
    ///   - rhs: A column type.
    /// - Returns: A Boolean array.
    public static func <= (lhs: Self.Element, rhs: Self) -> Self.Bool

    /// Returns a Boolean array that indicates whether the corresponding element of a column type
    /// is greater than a value.
    ///   - lhs: A column type.
    ///   - rhs: A value of the same type as the column.
    /// - Returns: A Boolean array.
    public static func > (lhs: Self, rhs: Self.Element) -> Self.Bool

    /// Returns a Boolean array that indicates whether the value
    /// is greater than the corresponding element of a column type.
    ///   - lhs: A value of the same type as the column.
    ///   - rhs: A column type.
    /// - Returns: A Boolean array.
    public static func > (lhs: Self.Element, rhs: Self) -> Self.Bool

    /// Returns a Boolean array that indicates whether the corresponding element of a column type
    /// is greater than or equal to a value.
    ///   - lhs: A column type.
    ///   - rhs: A value of the same type as the column.
    /// - Returns: A Boolean array.
    public static func >= (lhs: Self, rhs: Self.Element) -> Self.Bool

    /// Returns a Boolean array that indicates whether the value
    /// is greater than or equal to the corresponding element of a column type.
    ///   - lhs: A value of the same type as the column.
    ///   - rhs: A column type.
    /// - Returns: A Boolean array.
    public static func >= (lhs: Self.Element, rhs: Self) -> Self.Bool

    /// Returns a Boolean array that indicates whether the corresponding element of a column type
    /// is equal to a value.
    ///   - lhs: A column type.
    ///   - rhs: A value of the same type as the column.
    /// - Returns: A Boolean array.
    public static func == (lhs: Self, rhs: Self.Element) -> Self.Bool
```

```
    /// Returns a Boolean array that indicates whether the value
    /// is equal to the corresponding element of a column type.
    ///   - lhs: A value of the same type as the column.
    ///   - rhs: A column type.
    /// - Returns: A Boolean array.
    public static func == (lhs: Self.Element, rhs: Self) -> Self.
Bool

    /// Returns a Boolean array that indicates whether the corresponding
element of a column type
    /// isn't equal to a value.
    ///   - lhs: A column type.
    ///   - rhs: A value of the same type as the column.
    /// - Returns: A Boolean array.
    public static func != (lhs: Self, rhs: Self.Element) -> Self.
Bool

    /// Returns a Boolean array that indicates whether the value
    /// isn't equal to the corresponding element of a column type.
    ///   - lhs: A value of the same type as the column.
    ///   - rhs: A column type.
    /// - Returns: A Boolean array.
    public static func != (lhs: Self.Element, rhs: Self) -> Self.
Bool


/// A collection that represents a selection of contiguous elements from a typed
column.
///
/// A column slice contains only certain elements from its parent column.
/// Create a slice by using a subscript with a range.
///
/// ```swift
/// let slice = column[100 ..< 200]
/// ```
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public struct ColumnSlice WrappedElement
OptionalColumnProtocol

    /// The type of the column slice's elements, which is an optional type of the
parent column's type.
    public typealias Element   WrappedElement

    /// The type that represents a position in the column slice.
    public typealias Index   Int

    /// The name of the slice's parent column.
    @inlinable public var name  String
```

```
    /// The underlying type of the column's elements.
    @inlinable public var wrappedElementType  any Any
get

    /// A prototype that creates type-erased columns with the same underlying
type as the column slice.
    ///
    /// Use a type-erased column prototype to create new columns of the same
type as the slice's parent column
    /// without explicitly knowing what type it is.
    public var prototype  any AnyColumnPrototype   get

    /// Creates a slice with the contents of a column.
    ///
    /// - Parameter column: A column.
    @inlinable public init _          Column WrappedElement

    /// Creates a new column by applying a transformation to every element.
    ///
    /// - Parameter transform: The transformation closure.
    @inlinable public func map T  _
 ColumnSlice WrappedElement  Element  throws    T   rethrows
   Column T

    /// Generates a slice that contains the elements that satisfy the predicate.
    ///
    /// - Parameter isIncluded: The filter predicate. Elements for which
the predicate returns `true` are included.
    public func filter _
 ColumnSlice WrappedElement  Element  throws    Bool  rethrows
   DiscontiguousColumnSlice WrappedElement

    /// Returns a type-erased column slice.
    public func eraseToAnyColumn      AnyColumnSlice

    /// A type that represents the indices that are valid for subscripting the
    /// collection, in ascending order.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias Indices
Range ColumnSlice WrappedElement  Index

    /// A type that provides the collection's iteration interface and
    /// encapsulates its iteration state.
    ///
    /// By default, a collection conforms to the `Sequence` protocol by
    /// supplying `IndexingIterator` as its associated `Iterator`
    /// type.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias Iterator
IndexingIterator ColumnSlice WrappedElement
```

```
    /// A collection representing a contiguous subrange of this collection's
    /// elements. The subsequence shares indices with the original collection.
    ///
    /// The default subsequence type for collections that don't define their own
    /// is `Slice`.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias SubSequence   ColumnSlice WrappedElement


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension ColumnSlice where WrappedElement   Hashable

    /// Generates a categorical summary of the column slice's elements.
    public func summary
CategoricalSummary WrappedElement


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension ColumnSlice   Sendable where WrappedElement
Sendable


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension ColumnSlice   RandomAccessCollection
MutableCollection

    /// The index of the initial element in the column slice.
    @inlinable public var startIndex  Int   get

    /// The index of the final element in the column slice.
    @inlinable public var endIndex  Int   get

    /// Returns the index immediately after an element index.
    /// - Parameter i: A valid index to an element in the column slice.
    @inlinable public func index          Int     Int

    /// Returns the index immediately before an element index.
    /// - Parameter i: A valid index to an element in the column slice.
    @inlinable public func index           Int     Int

    /// The number of elements in the column slice.
    @inlinable public var count  Int   get

    /// The number of missing elements in the column slice.
    @available macOS 12  iOS 15  tvOS 15  watchOS 8
    @inlinable public var missingCount  Int   get

    /// Accesses an element at an index.
```

```
    ///
    /// - Parameter position: A valid index to an element in the column
slice.
    @inlinable public subscript           Int
ColumnSlice WrappedElement  Element

    /// Returns a Boolean that indicates whether the element at an index is
missing.
    ///
    /// - Parameter index: An element index.
    public func isNil           Int     Bool

    /// Accesses a contiguous range of elements.
    ///
    /// - Parameter range: A range of valid indices in the column slice.
    @inlinable public subscript        Range Int
ColumnSlice WrappedElement


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension ColumnSlice   Equatable where WrappedElement
Equatable

    /// Returns a Boolean that indicates whether the column slices are equal.
    /// - Parameters:
    ///   - lhs: A typed column slice.
    ///   - rhs: Another typed column slice.
    public static func          ColumnSlice WrappedElement
     ColumnSlice WrappedElement      Bool


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension ColumnSlice   Hashable where WrappedElement
Hashable

    /// Hashes the essential components of the column slice by feeding them
into a hasher.
    /// - Parameter hasher: A hasher the method uses to combine the
components of the column slice.
    public func hash              inout Hasher

    /// Generates a discontiguous slice that contains unique elements.
    ///
    /// The method only adds the first of multiple elements with the same value
    /// --- the element with the smallest index ---
    /// to the slice.
    ///
    /// - Returns: A discontiguous column slice.
    public func distinct
DiscontiguousColumnSlice WrappedElement
```

```
    /// The hash value.
    ///
    /// Hash values are not guaranteed to be equal across different executions of
    /// your program. Do not save hash values to use during a future execution.
    ///
    /// - Important: `hashValue` is deprecated as a `Hashable` requirement. To
    ///   conform to `Hashable`, implement the `hash(into:)` requirement instead.
    ///   The compiler provides an implementation for `hashValue` for you.
    public var hashValue: Int { get }


@available(macOS 12, iOS 15, tvOS 15, watchOS 8)
extension ColumnSlice

    /// Modifies a column slice by adding a value to each element.
    ///
    /// - Parameters:
    ///   - lhs: A column slice.
    ///   - rhs: A value of the same type as the column's elements.
    public static func += (lhs: inout ColumnSlice<WrappedElement>, rhs: WrappedElement) where WrappedElement: AdditiveArithmetic

    /// Modifies a column slice by subtracting a value from each element.
    ///
    /// - Parameters:
    ///   - lhs: A column slice.
    ///   - rhs: A value of the same type as the column's elements.
    public static func -= (lhs: inout ColumnSlice<WrappedElement>, rhs: WrappedElement) where WrappedElement: AdditiveArithmetic

    /// Modifies a column slice by multiplying each element by a value.
    ///
    /// - Parameters:
    ///   - lhs: A column slice.
    ///   - rhs: A value of the same type as the column's elements.
    public static func *= (lhs: inout ColumnSlice<WrappedElement>, rhs: WrappedElement) where WrappedElement: Numeric

    /// Modifies an integer column slice by dividing each element by a value.
    ///
    /// - Parameters:
    ///   - lhs: A column slice.
    ///   - rhs: A value of the same type as the column's elements.
    public static func /= (lhs: inout ColumnSlice<WrappedElement>, rhs: WrappedElement)
```

```
ColumnSlice WrappedElement        WrappedElement  where
WrappedElement   BinaryInteger

    /// Modifies a floating-point column slice by dividing each element by a value.
    ///
    /// - Parameters:
    ///   - lhs: A column slice.
    ///   - rhs: A value of the same type as the column's elements.
    public static func          inout
ColumnSlice WrappedElement        WrappedElement  where
WrappedElement   FloatingPoint


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension ColumnSlice

    /// Modifies a column slice by adding each value in a collection to
    /// the corresponding element in the column.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the
column's elements.
    public static func     C       inout
ColumnSlice WrappedElement        C  where WrappedElement
AdditiveArithmetic  WrappedElement    C Element  C
Collection

    /// Modifies a column slice by adding each optional value in a collection to
    /// the corresponding element in the column.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the
column's elements.
    public static func     C       inout
ColumnSlice WrappedElement        C  where WrappedElement
AdditiveArithmetic  C   Collection  C Element
WrappedElement

    /// Modifies a column slice by subtracting each value in a collection from
    /// the corresponding element in the column.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the
column's elements.
    public static func     C       inout
ColumnSlice WrappedElement        C  where WrappedElement
AdditiveArithmetic  WrappedElement    C Element  C
```

```
Collection

    /// Modifies a column slice by subtracting each optional value in a collection
from
    /// the corresponding element in the column.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the
column's elements.
    public static func     C       inout
ColumnSlice WrappedElement        C  where WrappedElement
AdditiveArithmetic  C   Collection  C Element
WrappedElement

    /// Modifies a column slice by multiplying each element in the column by
    /// the corresponding value in a collection.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the
column's elements.
    public static func     C       inout
ColumnSlice WrappedElement        C  where WrappedElement
Numeric  WrappedElement    C Element  C   Collection

    /// Modifies a column slice by multiplying each element in the column by
    /// the corresponding optional value in a collection.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the
column's elements.
    public static func     C       inout
ColumnSlice WrappedElement        C  where WrappedElement
Numeric  C   Collection  C Element    WrappedElement

    /// Modifies an integer column slice by dividing each element in the column
by
    /// the corresponding value in a collection.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the
column's elements.
    public static func     C       inout
ColumnSlice WrappedElement        C  where WrappedElement
BinaryInteger  WrappedElement    C Element  C   Collection

    /// Modifies an integer column slice by dividing each element in the column
```

```
by
    /// the corresponding optional value in a collection.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the
column's elements.
    public static func  C  inout
ColumnSlice WrappedElement  C  where WrappedElement
BinaryInteger  C  Collection  C Element  WrappedElement

    /// Modifies a floating-point column slice by dividing each element in the
column by
    /// the corresponding value in a collection.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the
column's elements.
    public static func  C  inout
ColumnSlice WrappedElement  C  where WrappedElement
FloatingPoint  WrappedElement  C Element  C  Collection

    /// Modifies a floating-point column slice by dividing each element in the
column by
    /// the corresponding optional value in a collection.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the
column's elements.
    public static func  C  inout
ColumnSlice WrappedElement  C  where WrappedElement
FloatingPoint  C  Collection  C Element  WrappedElement


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension ColumnSlice where WrappedElement  Comparable

    /// Returns the element with the lowest value, ignoring missing elements.
    public func min  ColumnSlice WrappedElement  Element

    /// Returns the element with the highest value, ignoring missing elements.
    public func max  ColumnSlice WrappedElement  Element

    /// Returns the index of the element with the lowest value, ignoring missing
elements.
    public func argmin  Int

    /// Returns the index of the element with the highest value, ignoring missing
```

```
elements.
    public func argmax      Int


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension ColumnSlice where WrappedElement
AdditiveArithmetic

    /// Returns the sum of the column slice's elements, ignoring missing
elements.
    public func sum      WrappedElement


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension ColumnSlice where WrappedElement   FloatingPoint

    /// Returns the mean average of the floating-point slice's elements, ignoring
missing elements.
    public func mean      ColumnSlice WrappedElement  Element

    /// Returns the standard deviation of the floating-point column slice's
elements, ignoring missing elements.
    ///
    /// - Parameter deltaDegreesOfFreedom: A nonnegative integer.
    /// The method calculates the standard deviation's divisor by subtracting this
parameter from the number of
    /// non-`nil` elements (`n - deltaDegreesOfFreedom` where `n` is
the number of non-`nil` elements).
    ///
    /// - Returns: The standard deviation; otherwise, `nil` if there are
fewer than
    /// `deltaDegreesOfFreedom + 1` non-`nil` items in the column.
    public func standardDeviation                          Int
1     ColumnSlice WrappedElement  Element


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension ColumnSlice where WrappedElement   BinaryInteger

    /// Returns the mean average of the integer slice's elements, ignoring
missing elements.
    public func mean      Double

    /// Returns the standard deviation of the integer column slice's elements,
ignoring missing elements.
    ///
    /// - Parameter deltaDegreesOfFreedom: A nonnegative integer.
    /// The method calculates the standard deviation's divisor by subtracting this
parameter from the number of
    /// non-`nil` elements (`n - deltaDegreesOfFreedom` where `n` is
the number of non-`nil` elements).
```

```
    ///
    /// - Returns: The standard deviation; otherwise, `nil` if there are
fewer than
    /// `deltaDegreesOfFreedom + 1` non-`nil` items in the column.
    public func standardDeviation                              Int
1     Double


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension ColumnSlice   CustomStringConvertible
CustomDebugStringConvertible  CustomReflectable

    /// A text representation of the column slice.
    public var description  String   get

    /// A text representation of the column slice suitable for debugging.
    public var debugDescription  String   get

    /// A mirror that reflects the column slice.
    public var customMirror  Mirror   get


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension ColumnSlice where WrappedElement
BinaryFloatingPoint

    /// Generates a numeric summary of the floating-point column slice's
elements.
    public func numericSummary
NumericSummary WrappedElement


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension ColumnSlice where WrappedElement   BinaryInteger

    /// Generates a numeric summary of the integer column slice's elements.
    public func numericSummary      NumericSummary Double


/// A collection that arranges data in rows and columns.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
@dynamicMemberLookup public struct DataFrame
DataFrameProtocol

    /// The entire data frame as a collection of columns.
    public var columns    AnyColumn    get

    /// The entire data frame as a collection of rows.
    public var rows  DataFrame Rows
```

```
    /// The number of rows and columns in the data frame.
    ///
    /// - Parameters:
    ///   - rows: The number of rows in the data frame.
    ///   - columns: The number of columns in the data frame.
    public var shape         Int           Int    get

    /// Creates an empty data frame with no rows or columns.
    public init

    /// Creates a new data frame from a sequence of columns.
    /// - Parameter columns: A sequence of type-erased columns.
    public init S           S  where S   Sequence  S Element
   AnyColumn

    /// Creates a new data frame with a slice of rows from another data frame.
    /// - Parameter other: A row slice from another data frame.
    public init _        DataFrame Slice

    /// Returns the index of a column.
    ///
    /// - Parameter columnName: The name or an alias of the column.
    /// - Returns: An integer if the column name or alias exists in the data
frame;
    ///   otherwise, `nil`.
    ///
    ///   This method's complexity is O(*1*).
    public func indexOfColumn _             String     Int

    /// Returns a Boolean value indicating whether the data frame contains a
column matching a column ID.
    ///
    /// - Parameter id: A column ID.
    @available macOS 12.3  iOS 15.4  tvOS 15.4  watchOS 8.5

    @inlinable public func containsColumn T  _
ColumnID T      Bool

    /// Returns a Boolean value indicating whether the data frame contains a
column.
    ///
    /// - Parameters:
    ///   - name: A column name.
    ///   - type: An element type.
    @available macOS 12.3  iOS 15.4  tvOS 15.4  watchOS 8.5

    @inlinable public func containsColumn T  _       String  _
      T          Bool

    /// Returns a Boolean value indicating whether the data frame contains a
```

```
column.
    ///
    /// - Parameters:
    ///   - name: A column name.
    @available macOS 13.0  iOS 16.0  tvOS 16.0  watchOS 9.0

    @inlinable public func containsColumn _       String
Bool

    /// Returns the column names for an alias.
    ///
    /// Use this method to discover whether an alias refers to more than one
column.
    /// For example, a data frame may have multiple columns with the same
name
    /// after you call its ``joined(_:on:kind:)-6moa8`` method.
    public func columnNames                   String
 String

    /// Adds an alternative name for a column.
    /// - Parameters:
    ///   - alias: An additional name for the column.
    ///   - columnName: The name of a column.
    public mutating func addAlias _        String
           String

    /// Removes an alternative name for a column.
    /// - Parameter alias: An additional name for the column.
    public mutating func removeAlias _        String

    /// Adds a typed column to the end of the data frame.
    /// - Parameter column: A typed column.
    /// The column must have the same number of rows as the data frame,
    /// and must not have the same name as another column in the data frame.
    ///
    /// The column you append becomes the last column and has the highest
index
    /// in the data frame.
    public mutating func append T          Column T

    /// Adds a type-erased column to the end of the data frame.
    /// - Parameter column: A type-erased column.
    /// The column must have the same number of rows as the data frame,
    /// and must not have the same name as another column in the data frame.
    public mutating func append          AnyColumn

    /// Adds a typed column at a position in the data frame.
    /// - Parameters:
    ///   - column: A typed column.
    ///     The column must have the same number of rows as the data frame,
```

```
    ///       and must not have the same name as another column in the data
frame.
    ///
    ///   - index: A column position in the data frame.
    ///
    /// The method inserts the new column before the column currently at
`index`.
    /// If you pass the array's `shape.columns` property as the `index`
parameter,
    /// the method appends the new column to the data frame.
    public mutating func insert T          Column T
       Int

    /// Adds a type-erased column at a position in the data frame.
    /// - Parameters:
    ///   - column: A type-erased column.
    ///     The column must have the same number of rows as the data frame,
    ///     and must not have the same name as another column in the data
frame.
    ///
    ///   - index: A column position in the data frame.
    ///
    /// The method inserts the new column before the column currently at
`index`.
    /// If you pass the array's `shape.column` property as the `index`
parameter,
    /// the method appends the new column to the data frame.
    public mutating func insert          AnyColumn
Int

    /// Removes a column you select by its column identifier from the data frame.
    /// - Parameter id: The identifier of a column in the data frame.
    /// - Returns: The column the method removes from the data frame.
    @discardableResult
    public mutating func removeColumn T  _     ColumnID T
Column T

    /// Removes a column you select by its name from the data frame.
    /// - Parameter name: The name of a column in the data frame.
    /// - Returns: The column the method removes from the data frame.
    @discardableResult
    public mutating func removeColumn _       String
AnyColumn

    /// Applies a transform closure that modifies the elements of a column you
select by column identifier.
    /// - Parameters:
    ///   - id: The identifier of a column in the data frame.
    ///   - transform: A closure that transforms each element in the
column.
```

```
    public mutating func transformColumn From  To  _
ColumnID From   _             From   throws    To   rethrows

    /// Applies a transform closure that modifies the nonempty elements of a
column
    /// you select by column identifier.
    /// - Parameters:
    ///   - id: The identifier of a column in the data frame.
    ///   - transform: A closure that transforms each non-`nil` element
in the column.
    public mutating func transformColumn From  To  _
ColumnID From   _             From  throws    To   rethrows

    /// Applies a transform closure that modifies the elements of a column you
select by name.
    /// - Parameters:
    ///   - name: The name of a column in the data frame.
    ///   - transform: A closure that transforms each element in the
column.
    public mutating func transformColumn From  To  _
String  _             From   throws    To   rethrows

    /// Applies a transform closure that modifies the nonempty elements of a
column you select by name.
    /// - Parameters:
    ///   - name: The name of a column in the data frame.
    ///   - transform: A closure that transforms each element in the
column.
    public mutating func transformColumn From  To  _
String  _             From  throws    To   rethrows

    /// Adds a row of values to the data frame.
    /// - Parameter row: A row from a data frame.
    public mutating func append      DataFrame Row

    /// Adds a comma-separated, or variadic, list of values as a row to the data
frame.
    /// - Parameter row: A comma-separated, or variadic, list of optional
values.
    /// Each value's type must match the type of the corresponding column in the
data frame.
    public mutating func append      Any

    /// Adds a dictionary's values as a row to the data frame.
    /// - Parameter dictionary: A dictionary of values whose key is a
column's name.
    /// Each key in the dictionary must be the name or alias of a column in the
data frame.
    /// Each value in the dictionary must be of the same types as the
corresponding column.
    public mutating func append
```

```
 String   Any

    /// Adds an empty row to the data frame.
    ///
    /// Each value in an empty row is `nil`.
    public mutating func appendEmptyRow

    /// Adds a row of values at a position in the data frame.
    /// - Parameters:
    ///   - row: A row from a data frame.
    ///   - index: A row position in the data frame.
    /// The method inserts the new row before the row currently at `index`.
    /// If you pass the array's `shape.rows` property as the `index`
parameter,
    /// the method appends the new row to the data frame.
    public mutating func insert      DataFrame Row
Int

    /// Removes a row from the data frame.
    /// - Parameter index: A row position in the data frame.
    public mutating func removeRow           Int

    /// Returns a slice that contains the initial rows up to a maximum length.
    ///
    /// - Parameter maxLength: The maximum number of rows.
    public func prefix _            Int     DataFrame Slice

    /// Returns a slice that contains the final rows up to a maximum length.
    ///
    /// - Parameter maxLength: The maximum number of rows.
    public func suffix _            Int     DataFrame Slice

    /// Adds the rows of another data frame that has the same column names
and types.
    ///
    /// This method raises a fatal error if the data frame columns don't match.
    /// The following code shows how to check that the columns match.
    ///
    /// ```
    /// let sameColumns = left.columns.count ==
right.columns.count && zip(left.columns,
right.columns).allSatisfy {
    ///     $0.name == $1.name && $0.wrappedElementType ==
$1.wrappedElementType
    /// }
    /// ```
    ///
    /// - Parameter other: Another data frame that has the same number
of columns.
    ///   The columns in `other` must have the same names and types as the
```

columns in the data frame.

```
    public mutating func append                DataFrame

    /// Adds the rows of another data frame.
    ///
    /// The method ignores columns in `other` that don't exist in the data
frame.
    /// It fills the values for columns in the data frame that don't exist in `other`
to `nil`.
    /// It raises a fatal error if columns with the same name have different types.
    /// The following code shows how to check that the columns match.
    ///
    /// ```
    /// func columnTypesAreEqual(_ left: DataFrame, _ right:
DataFrame) -> Bool {
    ///     for rightColumn in right.columns {
    ///         guard let index =
left.indexOfColumn(rightColumn.name) else {
    ///             continue
    ///         }
    ///         let leftColumn = left.columns[index]
    ///         if leftColumn.wrappedElementType !=
rightColumn.wrappedElementType {
    ///             return false
    ///         }
    ///     }
    ///     return true
    /// }
    /// ```
    ///
    /// - Parameter other: Another data frame. The columns in `other`
that have the same name as columns in the data
    ///   frame must also have the same type.
    public mutating func append _        DataFrame

    /// Adds the rows of a slice from a data frame.
    /// - Parameter other: A slice of a data frame. The columns in
`other` that have
    /// the same name as columns in the data frame must also have the same
type.
    ///
    /// The method ignores columns in `other` that don't exist in the data
frame.
    /// The method fills the values for columns in the data frame that don't exist
in `other` to `nil`.
    public mutating func append _        DataFrame Slice

    /// Returns a selection of rows that satisfy a predicate in the columns you
select by name.
    /// - Parameters:
```

```
    ///   - columnName: The name of a column.
    ///   - type: The type of the column.
    ///   - isIncluded: A predicate closure that receives an element of the column as its argument
    ///     and returns a Boolean that indicates whether the slice includes the element's row.
    /// - Returns: A data frame slice that contains the rows that satisfy the predicate.
    public func filter<T>(_ columnName: String, _ type: T.Type, throws isIncluded: (T) throws -> Bool) rethrows -> DataFrame.Slice

    /// Returns a selection of rows that satisfy a predicate in the columns you select by column identifier.
    /// - Parameters:
    ///   - columnID: The identifier of a column in the data frame.
    ///   - isIncluded: A predicate closure that receives an element of the column as its argument
    ///     and returns a Boolean that indicates whether the slice includes the element's row.
    /// - Returns: A data frame slice that contains the rows that satisfy the predicate.
    public func filter<T>(_ columnID: ColumnID<T>, _ isIncluded: (T) throws -> Bool) rethrows -> DataFrame.Slice

    /// Returns a selection of rows that satisfy a predicate.
    /// - Parameter isIncluded: A predicate closure that receives an row and
    ///     returns a Boolean that indicates whether the slice includes that row.
    /// - Returns: A data frame slice that contains the rows that satisfy the predicate.
    public func filter(_ isIncluded: (DataFrame.Row) throws -> Bool) rethrows -> DataFrame.Slice

    /// Generates a data frame that includes the columns you select with a sequence of names.
    /// - Parameter columnNames: A sequence of column names.
    /// - Returns: A new data frame.
    public func selecting<S>(_ columnNames: S) -> DataFrame where S : Sequence, S.Element == String

    /// Generates a data frame that includes the columns you select with a list of names.
    /// - Parameter columnNames: A comma-separated, or variadic, list of column names.
    /// - Returns: A new data frame.
    public func selecting(_ columnNames: String...) -> DataFrame

    /// A type that conforms to the type-erased column protocol.
    @available(iOS 15, tvOS 15, watchOS 8, macOS 12, *)
    public typealias ColumnType = AnyColumn
```

```
extension DataFrame

    /// Renames a column in the data frame.
    ///
    /// - Parameters:
    ///   - name: The name of a column in the data frame.
    ///   - newName: The new name for the column. The new name must not be the
    ///   same as another column in the data frame.
    @available macOS 12  iOS 15  tvOS 15  watchOS 8
    public mutating func renameColumn _       String
         String

    /// Replaces a column in the data frame, by name, with a type-erased column.
    ///
    /// - Parameters:
    ///   - name: The name of a column in the data frame.
    ///   - newColumn: Another column that replaces the column.
    @available macOS 12  iOS 15  tvOS 15  watchOS 8
    public mutating func replaceColumn _       String
          AnyColumn

    /// Replaces a column in the data frame, by column identifier, with a type-erased column.
    ///
    /// - Parameters:
    ///   - id: The identifier of a column in the data frame.
    ///   - newColumn: Another column that replaces the column.
    @available macOS 12  iOS 15  tvOS 15  watchOS 8
    public mutating func replaceColumn T  _     ColumnID T
               AnyColumn

    /// Replaces a column in the data frame, by name, with a typed column.
    ///
    /// - Parameters:
    ///   - name: The name of a column in the data frame.
    ///   - newColumn: Another column that replaces the column.
    @available macOS 12  iOS 15  tvOS 15  watchOS 8
    public mutating func replaceColumn T  _       String
          Column T

    /// Replaces a column in the data frame, by column identifier, with a typed column.
    ///
    /// - Parameters:
    ///   - id: The identifier of a column in the data frame.
    ///   - newColumn: Another column that replaces the column.
```

```
    @available macOS 12  iOS 15  tvOS 15  watchOS 8
    public mutating func replaceColumn T  U  _
ColumnID T                  Column U


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DataFrame

    /// Accesses a column by its name to support dynamic-member lookup.
    /// - Parameter columnName: The name of a column.
    public subscript                          String
AnyColumn

    /// Accesses a column by its name.
    /// - Parameter columnName: The name of a column.
    public subscript             String     AnyColumn

    /// Accesses a column as an array by its name.
    ///
    /// - Parameters:
    ///   - columnName: The name of a column.
    ///   - type: The type of the column.
    public subscript T              String        T
  self      T

    /// Accesses a column by its name and type.
    /// - Parameters:
    ///   - columnName: The name of a column.
    ///   - type: The type of the column.
    public subscript T              String        T
Column T

    /// Accesses a column by its column identifier.
    /// - Parameter id: The identifier of a column.
    public subscript T      ColumnID T      Column T

    /// Generates a data frame that includes the columns in a sequence of
column names.
    /// - Parameter columnNames: A sequence of column names.
    /// - Returns: A new data frame.
    public subscript S               S     DataFrame where S
Sequence  S Element    String   get

    /// Accesses a column by its index.
    /// - Parameter index: The index of a column.
    public subscript               Int     AnyColumn

    /// Accesses a column by its index and type.
    /// - Parameters:
```

```
    ///   - index: The index of a column.
    ///   - type: The type of the column.
    public subscript<T>(index: Int, type: T.Type) -> Column<T>

    /// Accesses a row by its index.
    /// - Parameter index: The index of a row.
    public subscript(index: Int) -> DataFrame.Row

    /// Returns a slice of the rows by masking its elements with a Boolean column.
    /// - Parameter mask: A Boolean column that indicates whether the method includes a row in the slice.
    public subscript<C>(mask: C) -> DataFrame.Slice where C: Collection, C.Element == Bool { get }


@available(macOS 12, iOS 15, tvOS 15, watchOS 8, *)
extension DataFrame: Hashable {

    /// Returns a Boolean that indicates whether the data frames are equal.
    /// - Parameters:
    ///   - lhs: A data frame.
    ///   - rhs: Another data frame.
    public static func ==(lhs: DataFrame, rhs: DataFrame) -> Bool

    /// Hashes the essential components of the data frame by feeding them into a hasher.
    /// - Parameter hasher: A hasher the method uses to combine the components of the data frame.
    public func hash(into hasher: inout Hasher)

    /// The hash value.
    ///
    /// Hash values are not guaranteed to be equal across different executions of
    /// your program. Do not save hash values to use during a future execution.
    ///
    /// - Important: `hashValue` is deprecated as a `Hashable` requirement. To
    ///   conform to `Hashable`, implement the `hash(into:)` requirement instead.
    ///   The compiler provides an implementation for `hashValue` for you.
    public var hashValue: Int { get }


@available(macOS 12, iOS 15, tvOS 15, watchOS 8, *)
extension DataFrame: ExpressibleByDictionaryLiteral {

    /// Creates a data frame from a dictionary literal.
```

```
    ///
    /// - Parameters:
    ///   - elements: A comma-separated, or variadic, list of tuples.
    ///     Each tuple consists of two elements:
    ///     * A string that represents a column's name
    ///     * An array that represents the elements for that column
    ///
    /// Don't call this initializer directly.
    /// The compiler calls it to create a data frame from a dictionary literal.
    /// You create a dictionary literal by enclosing a comma-separated list of key-value pairs in square brackets.
    ///
    /// For example, this line creates a data frame with two columns and four rows:
    /// ```swift
    /// let dataFrame: DataFrame = ["a": [1, 2, 3, 5], "b": [1.414, 2.718, 3.14, 6.28]]
    /// ```
    ///
    /// The initializer checks each column's elements and, if possible, defines the column's type to one of the
    /// following:
    /// - `Bool`
    /// - `Int`
    /// - `Float`
    /// - `Double`
    /// - `Date`
    /// - `String`
    /// - `Data`
    ///
    /// Otherwise, the data frame sets a column's type to `Any`.
    ///
    /// > Note: Use ``append(column:)-aema`` to add a column of a specific type.
    public init                                    String
 Any

    /// The key type of a dictionary literal.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias Key   String

    /// The value type of a dictionary literal.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias Value    Any


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DataFrame

    /// A set of a data frame's rows you create by using a method from a data
```

```
frame instance
    /// or another data frame slice.
    ///
    /// A slice is an arbitrary set of rows from a data frame.
    /// For example, a slice might contain rows 0–3, 5–9, and 101 from its
underlying data frame.
    @dynamicMemberLookup public struct Slice
DataFrameProtocol

        /// The underlying data frame.
        public var base  DataFrame   get

        /// The entire slice as a collection of rows.
        public var rows  DataFrame Rows

        /// The entire slice as a collection of columns.
        public var columns   AnyColumnSlice    get

        /// The number of rows and columns in the slice.
        ///
        /// - Parameters:
        ///   - rows: The number of rows in the slice.
        ///   - columns: The number of columns in the slice.
        public var shape          Int           Int    get

        /// A type that conforms to the type-erased column protocol.
        @available iOS 15  tvOS 15  watchOS 8  macOS 12
        public typealias ColumnType   AnyColumnSlice



@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DataFrame

    /// Generates a data frame that summarizes the columns of the data frame.
    public func summary      DataFrame

    /// Generates a data frame that summarizes the columns you select by
name.
    ///
    /// - Parameter columnNames: A comma-separated, or variadic, list of
column names in the data frame.
    public func summary                  String
DataFrame

    /// Generates a data frame that summarizes the columns you select by
index.
    ///
    /// - Parameter columnIndices: A comma-separated, or variadic, list
of column indices in the data frame.
```

```
    public func summary                              Int
DataFrame


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DataFrame

    /// Merges two columns that you select by name into a new column.
    ///
    /// - Parameters:
    ///   - columnName1: The name of a column.
    ///   - columnName2: The name of another column.
    ///   - newColumnName: The name of the new column that replaces the
two columns.
    ///   - transform: A closure that combines the corresponding elements
of the two columns into one element.
    ///
    ///   The merged column replaces the original column.
    public mutating func combineColumns E1  E2  R  _
          String  _              String
            String             E1   E2   throws    R
rethrows

    /// Merges two columns that you select by column identifier into a new
column.
    ///
    /// - Parameters:
    ///   - columnID1: The identifier of a column.
    ///   - columnID2: The identifier of another column.
    ///   - newColumnName: The name of the new column that replaces the
two columns.
    ///   - transform: A closure that combines the corresponding elements
of the two columns into one element.
    ///
    ///   The merged column replaces the original column.
    public mutating func combineColumns E1  E2  R  _
         ColumnID E1   _            ColumnID E2
            String             E1   E2   throws    R
rethrows

    /// Merges three columns that you select by name into a new column.
    ///
    /// - Parameters:
    ///   - columnName1: The name of a column.
    ///   - columnName2: The name of a second column.
    ///   - columnName3: The name of a third column.
    ///   - newColumnName: The name of the new column that replaces the
three columns.
    ///   - transform: A closure that combines the corresponding elements
of the three columns into one element.
```

```
    ///
    ///    The merged column replaces the original column.
    public mutating func combineColumns E1  E2  E3  R  _
            String  _                String  _
String                      String              E1   E2   E3
throws    R   rethrows

    /// Merges three columns that you select by column identifier into a new
column.
    ///
    /// - Parameters:
    ///    - columnID1: The identifier of a column.
    ///    - columnID2: The identifier of a second column.
    ///    - columnID3: The identifier of a third column.
    ///    - newColumnName: The name of the new column that replaces the
three columns.
    ///    - transform: A closure that combines the corresponding elements
of the three columns into one element.
    ///
    ///    The merged column replaces the original column.
    public mutating func combineColumns E1  E2  E3  R  _
           ColumnID E1   _            ColumnID E2   _
           ColumnID E3                       String
            E1   E2   E3   throws    R   rethrows


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DataFrame

    /// Creates a data frame from a CSV file.
    ///
    /// - Parameters:
    ///    - url: A URL for a CSV file.
    ///    - columns: An array of column names; Set to `nil` to use every
column in the CSV file.
    ///    - rows: A range of indices; Set to `nil` to use every row in the
CSV file.
    ///    - types: A dictionary of column names and their CSV types.
    ///    The data frame infers the types for column names that aren't in the
dictionary.
    ///    - options: The options that tell the data frame how to read the CSV
file.
    /// - Throws: A `CSVReadingError` instance.
    public init                          URL            String
  nil        Range Int     nil          String   CSVType
              CSVReadingOptions             throws

    /// Creates a data frame from CSV file data.
    ///
    /// - Parameters:
```

```
    ///   - data: The contents of a CSV file in a
    ///   <doc://com.apple.documentation/documentation/Foundation/Data>
instance.
    ///   - columns: An array of column names; Set to `nil` to use every
column in the CSV file.
    ///   - rows: A range of indices; Set to `nil` to use every row in the
CSV file.
    ///   - types: A dictionary of column names and their CSV types.
    ///   The data frame infers the types for column names that aren't in the
dictionary.
    ///   - options: The options that tell the data frame how to read the CSV
data.
    /// - Throws: A `CSVReadingError` instance.
    public init                Data            String     nil
      Range Int     nil          String   CSVType
         CSVReadingOptions              throws


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DataFrame

    /// The underlying data frame.
    ///
    /// For a ``DataFrame`` instance, this property is equivalent to `self`.
    public var base  DataFrame   get


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DataFrame

    /// Decodes the elements of a column you select by name.
    ///
    /// - Parameters:
    ///   - type: The type of the decodable value.
    ///   - columnName: The name of a column.
    ///   - decoder: A decoder that accepts the column's type.
    /// - Throws: `ColumnDecodingError` when the decoder fails to
decode a column element.
    public mutating func decode T  Decoder   _        T
                    String                  Decoder  throws
where T   Decodable  Decoder   TopLevelDecoder

    /// Decodes the elements of a column you select by column identifier.
    ///
    /// - Parameters:
    ///   - type: The type of the decodable value.
    ///   - id: A column identifier.
    ///   - decoder: A decoder that accepts the column's type.
    /// - Throws: `ColumnDecodingError` when the decoder fails to
decode a column element.
```

```
    public mutating func decode T  Decoder  _       T
             ColumnID Decoder Input                  Decoder
throws where T   Decodable  Decoder   TopLevelDecoder


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DataFrame

    /// Arranges the rows of a data frame according to a column that you select
by its name.
    /// - Parameters:
    ///   - columnName: The name of a column.
    ///   - order: A sorting order.
    ///
    /// This is a convenience method that only works for columns of the following
types:
    /// - <doc://com.apple.documentation/documentation/Swift/Bool>
    /// - <doc://com.apple.documentation/documentation/Swift/Int>
    /// - <doc://com.apple.documentation/documentation/Swift/Float>
    /// - <doc://com.apple.documentation/documentation/Swift/Double>
    /// - <doc://com.apple.documentation/documentation/Foundation/Date>
    ///
    /// > Note: Elements with a value of `nil` are less than all non-`nil`
values.
    public mutating func sort               String
Order

    /// Arranges the rows of a data frame according to a column that you select
by its name and type.
    /// - Parameters:
    ///   - columnName: The name of a column.
    ///   - type: The column's type.
    ///   - order: A sorting order.
    /// > Note: Elements with a value of `nil` are less than all non-`nil`
values.
    public mutating func sort T                String  _
      T             Order              where T   Comparable

    /// Arranges the rows of a data frame according to a column that you select
by its column identifier.
    /// - Parameters:
    ///   - columnID0: The identifier of a column.
    ///   - order: A sorting order.
    /// > Note: Elements with a value of `nil` are less than all non-`nil`
values.
    public mutating func sort T              ColumnID T
   Order               where T   Comparable

    /// Arranges the rows of a data frame according to two columns that you
select by their column identifiers.
```

```
    /// - Parameters:
    ///   - columnID0: The identifier of a column.
    ///   - columnID1: The identifier of another column.
    ///   - order: A sorting order.
    /// > Note: Elements with a value of `nil` are less than all non-`nil`
values.
    public mutating func sort T0  T1
ColumnID T0              _              ColumnID T1          Order
             where T0   Comparable  T1   Comparable


    /// Arranges the rows of a data frame according to three columns that you
select by their column identifiers.
    /// - Parameters:
    ///   - columnID0: The identifier of a column.
    ///   - columnID1: The identifier of a second column.
    ///   - columnID2: The identifier of a third column.
    ///   - order: A sorting order.
    /// > Note: Elements with a value of `nil` are less than all non-`nil`
values.
    public mutating func sort T0  T1  T2
ColumnID T0   _              ColumnID T1
_
ColumnID T2          Order                 where T0
Comparable  T1   Comparable  T2   Comparable


    /// Arranges the rows of a data frame according to a column that you select
by its column identifier,
    /// with a predicate.
    /// - Parameters:
    ///   - columnID: The identifier of a column.
    ///   - areInIncreasingOrder: A closure that returns a Boolean that
indicates
    ///     whether the two elements are in increasing order.
    /// > Note: Elements with a value of `nil` are less than all non-`nil`
values.
    public mutating func sort T                ColumnID T
                      T  T  throws    Bool  rethrows


    /// Arranges the rows of a data frame according to a column that you select
by its name and type,
    /// with a predicate.
    /// - Parameters:
    ///   - columnName: The name of a column.
    ///   - type: The column's type.
    ///   - areInIncreasingOrder: A closure that returns a Boolean that
indicates
    ///     whether the two elements are in increasing order.
    /// > Note: Elements with a value of `nil` are less than all non-`nil`
values.
    public mutating func sort T                  String
_
      T                                  T  T  throws    Bool
rethrows
```

```
@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DataFrame   CustomStringConvertible
CustomDebugStringConvertible  CustomReflectable

    /// A text representation of the data frame.
    public var description  String   get

    /// A text representation of the data frame suitable for debugging.
    public var debugDescription  String   get

    /// A mirror that reflects the data frame.
    public var customMirror  Mirror   get


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DataFrame

    /// A single row within a data frame.
    public struct Row

        /// The row's underlying data frame.
        public var base  DataFrame   get

        /// The row's index in the underlying data frame.
        public let index  Int

        /// Accesses a value in the row you select by a column index and type.
        /// - Parameters:
        ///   - position: A valid column index in the row.
        ///   - type: The type of the column.
        public subscript T            Int       T          T

        /// Accesses a value in the row you select by a column name and type.
        /// - Parameters:
        ///   - columnName: The name of a column.
        ///   - type: The type of the column.
        public subscript T             String       T
    T

        /// Accesses a value in the row you select by a column name.
        /// - Parameter columnName: The name of a column.
        public subscript             String     Any

        /// Accesses a value in the row you select by a column identifier.
        /// - Parameter columnID: The identifier of a column.
        public subscript T            ColumnID T      T
```

```
@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DataFrame

    /// Creates a data frame by reading a JSON file.
    ///
    /// The JSON file should contain a sequence of objects where each object
contains a value for every column name.
    /// Here's an example with two columns "id" and "name":
    ///
    ///     [
    ///       {"id": 1, "name": "foo"},
    ///       {"id": 2, "name": "bar"},
    ///     ]
    ///
    /// - Parameters:
    ///   - url: A URL to a JSON file.
    ///   - columns: An array of column names; Set to `nil` to use every
column in the JSON file.
    ///   - types: A dictionary of column names and their JSON types.
    ///   The data frame infers the types for column names that aren't in the
dictionary.
    ///   - options: The options that instruct how the data frame reads the
JSON file.
    /// - Throws: A `JSONReadingError` instance.
    public init                                URL
 String     nil          String   JSONType
JSONReadingOptions            throws

    /// Creates a data frame by converting JSON data.
    ///
    /// The JSON file should contain a sequence of objects where each object
contains a value for every column name.
    /// Here's an example with two columns "id" and "name":
    ///
    ///     [
    ///       {"id": 1, "name": "foo"},
    ///       {"id": 2, "name": "bar"},
    ///     ]
    ///
    /// - Parameters:
    ///   - data: The contents of a JSON file as data.
    ///   - columns: An array of column names; Set to `nil` to use every
column in the JSON file.
    ///   - types: A dictionary of column names and their JSON types.
    ///   The data frame infers the types for column names that aren't in the
dictionary.
    ///   - options: The options that instruct how the data frame reads the
JSON file.
```

```
    /// - Throws: A `JSONReadingError` instance.
    public init(data: Data, options: String = nil) throws
        String: JSONType, JSONReadingOptions
```

```
@available(macOS 12, iOS 15, tvOS 15, watchOS 8)
extension DataFrame
```

```
    /// Encodes the elements of a column you select by name.
    ///
    /// - Parameters:
    ///   - columnName: The name of a column.
    ///   - type: The type of the column.
    ///   - encoder: A encoder that accepts the column's type.
    /// - Throws: `ColumnEncodingError` when the encoder fails to encode a column element.
    public mutating func encodeColumn<T: Encoder>(_: String, _: T.Type, encoder: T.Encoder) throws where T: Encodable, Encoder: TopLevelEncoder
```

```
    /// Encodes the elements of a column you select by column identifier.
    ///
    /// - Parameters:
    ///   - id: The name of a column.
    ///   - encoder: A encoder that accepts the column's type.
    /// - Throws: `ColumnEncodingError` when the encoder fails to encode a column element.
    public mutating func encodeColumn<T: Encoder, ColumnID: T>(_: ColumnID, _: T.Type, encoder: T.Encoder) throws where T: Encodable, Encoder: TopLevelEncoder
```

```
@available(macOS 12, iOS 15, tvOS 15, watchOS 8)
extension DataFrame
```

```
    /// Replaces each row in a collection column that you select by column identifier,
    /// with a new row for each element in the original row's collection.
    ///
    /// - Parameter id: A column identifier.
    public mutating func explodeColumn<ColumnID: T>(_: ColumnID) where T: Collection
```

```
    /// Replaces each row in a collection column that you select by name,
    /// with a new row for each element in the original row's collection.
    ///
    /// - Parameter name: A column name.
    public mutating func explodeColumn<T>(_: String, _: T.Type) where T: Collection
```

```
    /// Generates a data frame by replacing each row in a collection column that you select by name,
    /// with a new row for each element in the original row's collection.
    ///
    /// - Parameters:
    ///   - name: A column name.
    ///   - type: The underlying type of the column.
    public func explodingColumn<T>(_ name: String, _ type: T.Type) -> DataFrame where T: Collection

    /// Generates a data frame by replacing each row in a collection column that you select by column identifier,
    /// with a new row for each element in the original row's collection.
    ///
    /// - Parameter id: A column identifier.
    public func explodingColumn<T>(_ id: ColumnID<T>) -> DataFrame where T: Collection


@available(macOS 12, iOS 15, tvOS 15, watchOS 8, *)
extension DataFrame {

    /// Creates a data frame from a Turi Create scalable data frame.
    ///
    /// - Parameters:
    ///   - url: A URL to an `SFrame` directory.
    ///   - columns: An array of column names; Set to `nil` to use every column in the `SFrame`.
    ///   - rows: A range of indices; Set to `nil` to use every row in the `SFrame`.
    /// - Throws: An `SFrameReadingError` instance.
    public init(contentsOf url: URL, columns: [String]? = nil, rows: Range<Int>? = nil) throws


@available(macOS 12, iOS 15, tvOS 15, watchOS 8, *)
extension DataFrame {

    /// A collection of rows in a data frame.
    public struct Rows: BidirectionalCollection, MutableCollection {

        /// The index of the initial row in the collection.
        public var startIndex: Int { get }

        /// The index of the final row in the collection.
        public var endIndex: Int { get }

        /// The number of rows in the collection.
```

```
        public var count  Int   get

        /// Returns the row index immediately after a row index.
        /// - Parameter i: A valid index to a row in the collection.
        public func index          Int     Int

        /// Returns the row index immediately before a row index.
        /// - Parameter i: A valid index to a row in the collection.
        public func index           Int     Int

        /// Accesses a row at an index.
        /// - Parameter position: A valid index to a row in the collection.
        public subscript           Int     DataFrame Row

        /// Returns a row collection from an index range.
        /// - Parameter position: A valid index to a row in the collection.
        public subscript         Range Int      DataFrame Rows

        /// A type representing the sequence's elements.
        @available iOS 15  tvOS 15  watchOS 8  macOS 12
        public typealias Element   DataFrame Row

        /// A type that represents a position in the collection.
        ///
        /// Valid indices consist of the position of every element and a
        /// "past the end" position that's not valid for use as a subscript
        /// argument.
        @available iOS 15  tvOS 15  watchOS 8  macOS 12
        public typealias Index   Int

        /// A type that represents the indices that are valid for subscripting the
        /// collection, in ascending order.
        @available iOS 15  tvOS 15  watchOS 8  macOS 12
        public typealias Indices
DefaultIndices DataFrame Rows

        /// A type that provides the collection's iteration interface and
        /// encapsulates its iteration state.
        ///
        /// By default, a collection conforms to the `Sequence` protocol by
        /// supplying `IndexingIterator` as its associated `Iterator`
        /// type.
        @available iOS 15  tvOS 15  watchOS 8  macOS 12
        public typealias Iterator
IndexingIterator DataFrame Rows

        /// A collection representing a contiguous subrange of this collection's
        /// elements. The subsequence shares indices with the original
collection.
```

```
        ///
        /// The default subsequence type for collections that don't define their
own
        /// is `Slice`.
        @available iOS 15  tvOS 15  watchOS 8  macOS 12
        public typealias SubSequence   DataFrame Rows



@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DataFrame

    /// Creates a grouping of rows that the method selects
    /// by choosing unique values in a column.
    /// - Parameter columnName: The name of a column.
    public func grouped                 String     any
RowGroupingProtocol


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DataFrame Slice

    /// Returns a column you select by its name to support dynamic-member
lookup.
    /// - Parameter columnName: The name of a column.
    public subscript                              String
AnyColumnSlice   get

    /// Returns a column you select by its name.
    /// - Parameter columnName: The name of a column.
    public subscript             String     AnyColumnSlice
get

    /// Returns a column you select by its name and type.
    /// - Parameters:
    ///   - columnName: The name of a column.
    ///   - type: The type of the column.
    public subscript T              String        T
DiscontiguousColumnSlice T    get

    /// Returns a column you select by its column identifier.
    /// - Parameter id: The identifier of a column.
    public subscript T            ColumnID T
DiscontiguousColumnSlice T    get

    /// Returns a column you select by its index.
    /// - Parameter index: The index of a column.
    public subscript T                Int        T
DiscontiguousColumnSlice T    get
```

```
    /// Generates a data frame slice that includes the columns in a sequence of column names.
    ///
    /// - Parameter columnNames: A sequence of column names.
    /// - Returns: A new data frame slice.
    public subscript S                S     DataFrame Slice where S   Sequence  S Element    String   get

    /// Returns a selection of rows that satisfy a predicate in the columns you select by name.
    /// - Parameters:
    ///   - columnName: The name of a column.
    ///   - type: The type of the column.
    ///   - isIncluded: A predicate closure that receives an element of the column as its argument,
    ///     and returns a Boolean that indicates whether the slice includes the element's row.
    /// - Returns: A data frame slice that contains the rows that satisfy the predicate.
    public func filter T                  String  _ T       _                T   throws    Bool  rethrows DataFrame Slice

    /// Returns a selection of rows that satisfy a predicate in the columns you select by column identifier.
    /// - Parameters:
    ///   - columnID: The identifier of a column in the slice.
    ///   - isIncluded: A predicate closure that receives an element of the column as its argument,
    ///     and returns a Boolean that indicates whether the slice includes the element's row.
    /// - Returns: A data frame slice that contains the rows that satisfy the predicate.
    public func filter T                ColumnID T   _
             T   throws    Bool  rethrows    DataFrame Slice

    /// Returns a new slice that contains the initial elements of the original slice.
    ///
    /// - Parameter length: The number of elements in the new slice.
    /// The length must be greater than or equal to zero and less than or equal to the number of elements
    /// in the original slice.
    ///
    /// - Returns: A new slice of the underlying data frame.
    public func prefix _          Int     DataFrame Slice

    /// Returns a new slice that contains the initial elements of the original slice
    /// up to and including the element at a position.
    ///
    /// - Parameter position: A valid index to an element in the slice.
    ///
```

```
    /// - Returns: A new slice of the underlying data frame.
    public func prefix                    Int
DataFrame Slice

    /// Returns a new slice that contains the initial elements of the original slice
    /// up to, but not including, the element at a position.
    ///
    /// - Parameter position: A valid index to an element in the slice.
    ///
    /// - Returns: A new slice of the underlying data frame.
    public func prefix                  Int     DataFrame Slice

    /// Returns a new slice that contains the final elements of the original slice.
    ///
    /// - Parameter length: The number of elements in the new slice.
    /// The length must be greater than or equal to zero and less than or equal to
the number of elements
    /// in the original slice.
    ///
    /// - Returns: A new slice of the underlying data frame.
    public func suffix _          Int     DataFrame Slice

    /// Returns a new slice that contains the final elements of the original slice
    /// beginning with the element at a position.
    ///
    /// - Parameter position: A valid index to an element in the slice.
    ///
    /// - Returns: A new slice of the underlying data frame.
    public func suffix                  Int     DataFrame Slice

    /// Generates a data frame slice that includes the columns you select with a
sequence of names.
    ///
    /// - Parameter columnNames: A sequence of column names.
    /// - Returns: A new data frame slice.
    public func selecting S                S
DataFrame Slice where S   Sequence  S Element    String

    /// Generates a data frame slice that includes the columns you select with a
list of names.
    ///
    /// - Parameter columnNames: A comma-separated, or variadic, list of
column names.
    /// - Returns: A new data frame slice.
    public func selecting                 String
DataFrame Slice


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DataFrame Slice   Hashable
```

```
    /// Returns a Boolean that indicates whether the slices are equal.
    ///
    /// - Parameters:
    ///   - lhs: A data frame slice.
    ///   - rhs: Another data frame slice.
    public static func == (lhs: DataFrame.Slice, rhs: DataFrame.Slice) -> Bool

    /// Hashes the essential components of the data frame slice by feeding them into a hasher.
    /// - Parameter hasher: A hasher the method uses to combine the components of the slice.
    public func hash(into hasher: inout Hasher)

    /// The hash value.
    ///
    /// Hash values are not guaranteed to be equal across different executions of
    /// your program. Do not save hash values to use during a future execution.
    ///
    /// - Important: `hashValue` is deprecated as a `Hashable` requirement. To
    ///   conform to `Hashable`, implement the `hash(into:)` requirement instead.
    ///   The compiler provides an implementation for `hashValue` for you.
    public var hashValue: Int { get }


@available(macOS 12, iOS 15, tvOS 15, watchOS 8, *)
extension DataFrame.Slice {

    /// Generates a data frame that summarizes the columns of the data frame slice.
    public func summary() -> DataFrame

    /// Generates a data frame that summarizes the columns you select by name.
    ///
    /// - Parameter columnNames: A comma-separated, or variadic, list of column names in the data frame slice.
    public func summary(of columnNames: String...) -> DataFrame

    /// Generates a data frame that summarizes the columns you select by index.
    ///
    /// - Parameter columnIndices: A comma-separated, or variadic, list of column indices in the data frame slice.
    public func summary(of columnIndices: Int...) -> DataFrame
```

```
@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DataFrame Slice   CustomStringConvertible
CustomDebugStringConvertible  CustomReflectable

    /// A text representation of the data frame slice.
    public var description  String   get

    /// A text representation of the data frame slice suitable for debugging.
    public var debugDescription  String   get

    /// A mirror that reflects the data frame slice.
    public var customMirror  Mirror   get


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DataFrame Slice

    /// Creates a grouping of rows that the method selects
    /// by choosing unique values in a column.
    /// - Parameter columnName: The name of a column.
    public func grouped                String     any
RowGroupingProtocol


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DataFrame Row   CustomStringConvertible
CustomDebugStringConvertible  CustomReflectable

    /// A text representation of the row.
    public var description  String   get

    /// A text representation of the row.
    ///
    /// - Parameter options: A set of formatting options that affect the
description string.
    public func description          FormattingOptions
String

    /// A text representation of the row suitable for debugging.
    public var debugDescription  String   get

    /// A mirror that reflects the row.
    public var customMirror  Mirror   get


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DataFrame Row   RandomAccessCollection
MutableCollection
```

```
    /// The index of the initial column in the row.
    public var startIndex  Int   get

    /// The index of the final column in the row.
    public var endIndex  Int   get

    /// Returns the column index immediately before a column index in the row.
    /// - Parameter i: A valid column index to a value in the row.
    public func index          Int     Int

    /// Returns the column index immediately after a column index in the row.
    /// - Parameter i: A valid column index to a value in the row.
    public func index           Int     Int

    /// The number of columns in the row.
    public var count  Int   get

    /// Accesses a value at a column index.
    /// - Parameter position: A valid index to a column in the row.
    public subscript           Int     Any

    /// Accesses a slice from a range of indices.
    /// - Parameter bounds: A valid column index range.
    public subscript         Range Int
Slice DataFrame Row

    /// A type representing the sequence's elements.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias Element   Any

    /// A type that represents a position in the collection.
    ///
    /// Valid indices consist of the position of every element and a
    /// "past the end" position that's not valid for use as a subscript
    /// argument.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias Index   Int

    /// A type that represents the indices that are valid for subscripting the
    /// collection, in ascending order.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias Indices   Range Int

    /// A type that provides the collection's iteration interface and
    /// encapsulates its iteration state.
    ///
    /// By default, a collection conforms to the `Sequence` protocol by
    /// supplying `IndexingIterator` as its associated `Iterator`
```

```
    /// type.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias Iterator
IndexingIterator DataFrame Row

    /// A collection representing a contiguous subrange of this collection's
    /// elements. The subsequence shares indices with the original collection.
    ///
    /// The default subsequence type for collections that don't define their own
    /// is `Slice`.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias SubSequence   Slice DataFrame Row


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DataFrame Row   Hashable

    /// Returns a Boolean that indicates whether the rows are equal.
    /// - Parameters:
    ///   - lhs: A row.
    ///   - rhs: Another row.
    public static func          DataFrame Row
DataFrame Row     Bool

    /// Hashes the essential components of the row by feeding them into a
hasher.
    /// - Parameter hasher: A hasher the method uses to combine the
components of the row.
    public func hash              inout Hasher

    /// The hash value.
    ///
    /// Hash values are not guaranteed to be equal across different executions of
    /// your program. Do not save hash values to use during a future execution.
    ///
    /// - Important: `hashValue` is deprecated as a `Hashable`
requirement. To
    ///   conform to `Hashable`, implement the `hash(into:)` requirement
instead.
    ///   The compiler provides an implementation for `hashValue` for you.
    public var hashValue  Int   get


/// A type that represents a data frame.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public protocol DataFrameProtocol

    /// A type that conforms to the type-erased column protocol.
    associatedtype ColumnType   AnyColumnProtocol
```

```
    /// The underlying data frame.
    var base  DataFrame   get

    /// The rows of the underlying data frame.
    var rows  DataFrame Rows   get set

    /// The columns of the underlying data frame.
    var columns   Self ColumnType    get

    /// The number or rows and columns of the data frame type.
    /// - Parameters:
    ///   - rows: The number of rows in the data frame type.
    ///   - columns: The number of columns in the data frame type.
    var shape         Int           Int    get

    /// Accesses a slice of the data frame type with an index range.
    ///
    /// - Parameter range: An integer range.
    subscript        Range Int      DataFrame Slice   get
set


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DataFrameProtocol

    /// Generates two data frame slices by randomly splitting the rows of the data
table.
    /// - Parameters:
    ///   - proportion: A proportion in the range `[0.0, 1.0]`.
    ///   - seed: A seed number for a random-number generator.
    /// - Returns: A tuple of two data frame slices.
    public func randomSplit                Double        Int
  nil      DataFrame Slice  DataFrame Slice

    /// Generates two data frame slices by randomly splitting the rows of the data
table type
    /// with a random-number generator.
    /// - Parameters:
    ///   - proportion: A proportion in the range `[0.0, 1.0]`.
    ///   - generator: A random-number generator.
    /// - Returns: A tuple of two data frame slices.
    public func randomSplit G                  Double
          inout G      DataFrame Slice  DataFrame Slice
where G   RandomNumberGenerator

    /// Generates two data frames by randomly splitting the rows of a column,
    /// which you select by its name, into strata.
    ///
    /// - Parameters:
    ///   - columnName: The name of a column in the data frame type.
```

```
    ///   - proportion: A proportion in the range `[0.0, 1.0]`.
    ///   - randomSeed: A seed number for a random-number generator.
    ///
    /// - Returns: A tuple of two data frames.
    public func stratifiedSplit                    String
            Double              Int    nil      DataFrame
DataFrame

    /// Generates two data frames by randomly splitting the rows of multiple
columns,
    /// which you select by their names, into strata.
    ///
    /// - Parameters:
    ///   - columnNames: A comma-separated, or variadic, list of column
names.
    ///   - proportion: A proportion in the range `[0.0, 1.0]`.
    ///   - randomSeed: A seed number for a random-number generator.
    ///
    /// - Returns: A tuple of two data frames.
    public func stratifiedSplit                    String
            Double              Int    nil      DataFrame
DataFrame

    /// Generates two data frames by randomly splitting the rows of a column,
    /// which you select by column identifier,
    /// into strata.
    ///
    /// - Parameters:
    ///   - columnID: A column identifier.
    ///   - proportion: A proportion in the range `[0.0, 1.0]`.
    ///   - randomSeed: A seed number for a random-number generator.
    ///
    /// - Returns: A tuple of two data frames.
    public func stratifiedSplit T                ColumnID T
            Double              Int    nil      DataFrame
DataFrame  where T   Hashable

    /// Generates two data frames by randomly splitting the rows of two columns,
which you select by column identifiers,
    /// into strata.
    ///
    /// - Parameters:
    ///   - columnID0: A column identifier.
    ///   - columnID1: Another column identifier.
    ///   - proportion: A proportion in the range `[0.0, 1.0]`.
    ///   - randomSeed: A seed number for a random-number generator.
    ///
    /// - Returns: A tuple of two data frames.
    public func stratifiedSplit T0  T1
ColumnID T0   _            ColumnID T1
```

```
Double              Int     nil      DataFrame  DataFrame
where T0   Hashable  T1   Hashable

    /// Generates two data frames by randomly splitting the rows of three
columns,
    /// which you select by column identifiers, into strata.
    ///
    /// - Parameters:
    ///   - columnID0: A column identifier.
    ///   - columnID1: A second column identifier.
    ///   - columnID2: A third column identifier.
    ///   - proportion: A proportion in the range `[0.0, 1.0]`.
    ///   - randomSeed: A seed number for a random-number generator.
    ///
    /// - Returns: A tuple of two data frames.
    public func stratifiedSplit T0  T1  T2
ColumnID T0   _            ColumnID T1   _
ColumnID T2                  Double             Int    nil
    DataFrame  DataFrame  where T0   Hashable  T1   Hashable
T2   Hashable


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DataFrameProtocol

    /// Creates a CSV file with the contents of the data frame type.
    ///
    /// - Parameters:
    ///   - url: A location URL in the file system where the method saves the
CSV file.
    ///   - options: A ``CSVWritingOptions`` instance.
    public func writeCSV         URL
CSVWritingOptions            throws

    /// Generates a CSV data instance of the data frame type.
    ///
    /// - Parameters:
    ///   - options: A ``CSVWritingOptions`` instance.
    public func csvRepresentation          CSVWritingOptions
        throws    Data


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DataFrameProtocol

    /// A Boolean that indicates whether the data frame type is empty.
    public var isEmpty  Bool   get

    /// Accesses a slice of the data frame type with an index range.
    ///
```

```
    /// - Parameter range: An integer range.
    public subscript        Range Int      DataFrame Slice

    /// Accesses rows of a data frame type with an index range expression.
    ///
    /// - Parameter r: An integer range expression.
    @inlinable public subscript R     R     DataFrame Slice
where R   RangeExpression  R Bound    Int


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DataFrameProtocol

    /// Generates a data frame by copying the data frame's rows and then
sorting the rows according to a column
    /// that you select by its name.
    /// - Parameters:
    ///   - columnName: The name of a column.
    ///   - order: A sorting order.
    ///
    /// This is a convenience method that only works for columns of the following
types:
    /// - <doc://com.apple.documentation/documentation/Swift/Bool>
    /// - <doc://com.apple.documentation/documentation/Swift/Int>
    /// - <doc://com.apple.documentation/documentation/Swift/Float>
    /// - <doc://com.apple.documentation/documentation/Swift/Double>
    /// - <doc://com.apple.documentation/documentation/Foundation/Date>
    ///
    /// > Note: Elements with a value of `nil` are less than all non-`nil`
values.
    public func sorted                String         Order
                DataFrame

    /// Generates a data frame by copying the data frame's rows and then
sorting the rows according to a column
    /// that you select by its name and type.
    /// - Parameters:
    ///   - columnName: The name of a column.
    ///   - type: The column's type.
    ///   - order: A sorting order.
    /// > Note: Elements with a value of `nil` are less than all non-`nil`
values.
    public func sorted T                String
T             Order                DataFrame where T
Comparable

    /// Generates a data frame by copying the data frame's rows and then
sorting the rows according to a column
    /// that you select by its column identifier.
    /// - Parameters:
```

```
    ///     - columnID0: The identifier of a column.
    ///     - order: A sorting order.
    /// > Note: Elements with a value of `nil` are less than all non-`nil` values.
    public func sorted<T>(by columnID0: ColumnID<T>, order: Order<T> = .ascending) -> DataFrame where T: Comparable {

    /// Generates a data frame by copying the data frame's rows and then sorting the rows according to two columns
    /// that you select by their column identifiers.
    /// - Parameters:
    ///     - columnID0: The identifier of a column.
    ///     - columnID1: The identifier of another column.
    ///     - order: A sorting order.
    /// > Note: Elements with a value of `nil` are less than all non-`nil` values.
    public func sorted<T0, T1>(by columnID0: ColumnID<T0>, _ columnID1: ColumnID<T1>, order: Order<(T0, T1)> = .ascending) -> DataFrame where T0: Comparable, T1: Comparable {

    /// Generates a data frame by copying the data frame's rows and then sorting the rows according to three columns
    /// that you select by their column identifiers.
    /// - Parameters:
    ///     - columnID0: The identifier of a column.
    ///     - columnID1: The identifier of a second column.
    ///     - columnID2: The identifier of a third column.
    ///     - order: A sorting order.
    /// > Note: Elements with a value of `nil` are less than all non-`nil` values.
    public func sorted<T0, T1, T2>(by columnID0: ColumnID<T0>, _ columnID1: ColumnID<T1>, _ columnID2: ColumnID<T2>, order: Order<(T0, T1, T2)> = .ascending) -> DataFrame where T0: Comparable, T1: Comparable, T2: Comparable {

    /// Generates a data frame by copying the data frame's rows and then sorting the rows according to a column
    /// that you select by its name and type, with a predicate.
    /// - Parameters:
    ///     - columnName: The name of a column.
    ///     - type: The column's type.
    ///     - areInIncreasingOrder: A closure that returns a Boolean that indicates
    ///     whether the two elements are in increasing order.
    /// > Note: Elements with a value of `nil` are less than all non-`nil` values.
    public func sorted<T>(by columnName: String, _ type: T.Type, _ areInIncreasingOrder: (T, T) throws -> Bool) rethrows -> DataFrame
```

```
    /// Generates a data frame by copying the data frame's rows and then sorting the rows according to a column
    /// that you select by its column identifier, with a predicate.
    /// - Parameters:
    ///   - columnID: The identifier of a column.
    ///   - areInIncreasingOrder: A closure that returns a Boolean that indicates
    ///     whether the two elements are in increasing order.
    /// > Note: Elements with a value of `nil` are less than all non-`nil` values.
    public func sorted<T>(_ columnID: ColumnID<T>, _ areInIncreasingOrder: (T, T) throws -> Bool) rethrows -> DataFrame


@available(macOS 12, iOS 15, tvOS 15, watchOS 8)
extension DataFrameProtocol

    /// Generates a text representation of the data frame type.
    ///
    /// `FormattingOptions.maximumLineWidth` needs to be wide enough to print at least the index column, the truncation
    /// column, and one data column (at least two characters, one for initial of the content, and one for "…").
    ///
    /// - Parameter options: A set of formatting options that affect the description string,
    /// including the maximum width of a column and the maximum number of rows.
    public func description(options: FormattingOptions) -> String


@available(macOS 13, iOS 16, tvOS 16, watchOS 9)
extension DataFrameProtocol

    /// Creates a JSON file with the contents of the data frame.
    ///
    /// - Parameters:
    ///   - url: A location URL in the file system where the method saves the JSON file.
    ///   - options: A ``JSONWritingOptions`` instance.
    public func writeJSON(to url: URL, options: JSONWritingOptions) throws

    /// Generates a JSON data instance of the data frame.
    ///
    /// - Parameters:
    ///   - options: A ``JSONWritingOptions`` instance.
    public func jsonRepresentation(options: JSONWritingOptions)
```

```
            throws    Data


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DataFrameProtocol

    /// Creates a grouping of rows that the method selects
    /// by choosing unique values in a column.
    /// - Parameter columnID: A column identifier.
    ///
    /// - Returns: A collection of groups.
    public func grouped GroupingKey
ColumnID GroupingKey      RowGrouping GroupingKey  where
GroupingKey   Hashable

    /// Creates a grouping of rows that the method selects
    /// by choosing unique values the transform closure creates with elements of
a
    /// column you select by name.
    ///
    /// Create groupings that group rows by:
    /// * Telephone area codes
    /// * The first letter of a person's last name
    /// * A date's year or quarter
    /// * Number ranges
    ///
    /// - Parameters:
    ///   - columnName: The name of a column.
    ///   - transform: A closure that transforms a column's elements into a
hashable type.
    ///
    /// - Returns: A collection of groups.
    public func grouped InputKey  GroupingKey
String               InputKey      GroupingKey
RowGrouping GroupingKey  where GroupingKey   Hashable

    /// Creates a grouping of rows that the method selects
    /// by choosing unique values the transform closure creates with elements of
a
    /// column you select by column identifier.
    ///
    /// Create groupings that group rows by:
    /// * Telephone area codes
    /// * The first letter of a person's last name
    /// * A date's year or quarter
    /// * Number ranges
    ///
    /// - Parameters:
    ///   - columnID: A column identifier.
    ///   - transform: A closure that transforms a column's elements into a
```

```
hashable type.
    ///
    /// - Returns: A collection of groups.
    public func grouped<InputKey, GroupingKey>
ColumnID<InputKey>) -> InputKey.RowGrouping<GroupingKey>
RowGrouping<GroupingKey> where GroupingKey: Hashable

    /// Creates a grouping of rows that the method selects
    /// by choosing unique units of time in a date column you select by name.
    ///
    /// - Parameters:
    ///   - columnName: The name of a column.
    ///   - timeUnit: A component of a calendar date.
    ///
    /// - Returns: A collection of groups.
    ///
    /// After the method aggregates the groups, it creates a column with the
same name as the original column
    /// plus the `timeUnit` name.
    public func grouped(_ String,
Calendar.Component) -> RowGrouping<Int>

    /// Creates a grouping of rows that the method selects
    /// by choosing unique units of time in a date column you select by column
identifier.
    ///
    /// - Parameters:
    ///   - columnID: A column identifier.
    ///   - timeUnit: A component of a calendar date.
    ///
    /// - Returns: A collection of groups.
    ///
    /// After the method aggregates the groups, it creates a column with the
same name as the original column
    /// plus the `timeUnit` name.
    public func grouped(_ ColumnID<Date>,
Calendar.Component) -> RowGrouping<Int>

    /// Creates a grouping from multiple columns you select by name.
    ///
    /// - Parameters:
    ///   - columnNames: A comma-separated, or variadic, list of column
names.
    public func grouped(_ String, _ some
RowGroupingProtocol...)


    /// Creates a grouping from multiple columns that you select by column
identifier.
    ///
```

```
    /// - Parameters:
    ///   - columnIDs: A comma-separated, or variadic, list of column identifiers.
    public func grouped T                ColumnID T
some RowGroupingProtocol where T   Hashable


    /// Creates a grouping from two columns of different types.
    ///
    /// - Parameters:
    ///   - column0: A column identifier.
    ///   - column1: A second column identifier.
    public func grouped T0  T1              ColumnID T0   _
         ColumnID T1      some RowGroupingProtocol where T0
Hashable  T1   Hashable


    /// Creates a grouping from three columns of different types.
    ///
    /// - Parameters:
    ///   - column0: A column identifier.
    ///   - column1: A second column identifier.
    ///   - column2: A third column identifier.
    public func grouped T0  T1  T2              ColumnID T0
_          ColumnID T1   _           ColumnID T2      some
RowGroupingProtocol where T0   Hashable  T1   Hashable  T2
Hashable



@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DataFrameProtocol

    /// Generates a data frame by joining with another data frame type with a common column you select by name.
    ///
    /// - Parameters:
    ///   - other: A data frame type that represents the right side of the join.
    ///   - columnName: A column name that exists in both data frame types.
    ///   - kind: A join operation type.
    /// - Returns: A new data frame.
    public func joined R  _        R                String
      JoinKind             DataFrame where R
DataFrameProtocol

    /// Generates a data frame by joining with another data frame type with a common column
    /// that you select by identifier.
    ///
    /// - Parameters:
```

```
    ///   - other: A data frame type that represents the right side of the join.
    ///   - columnID: A column identifier that exists in both data frame types.
    ///   - kind: A join operation type.
    /// - Returns: A new data frame.
    public func joined R  T  _        R
ColumnID T         JoinKind             DataFrame where R
DataFrameProtocol  T   Hashable

    /// Generates a data frame by joining with another data frame type along
    /// the columns that you select by name for both data frame types.
    ///
    /// - Parameters:
    ///   - other: A data frame type that represents the right side of the join.
    ///   - columnNames: The column names of the data frame and the other
data frame type, `other`, respectively.
    ///   - kind: A join operation type.
    /// - Returns: A new data frame.
    public func joined R  _        R
String         String         JoinKind             DataFrame
where R   DataFrameProtocol

    /// Generates a data frame by joining with another data frame type along
    /// the columns that you select by identifier for both data frame types.
    ///
    /// - Parameters:
    ///   - other: A data frame type that represents the right side of the join.
    ///   - columnIDs: The column identifiers of the data frame and the
other data frame type, `other`, respectively.
    ///   - kind: A join operation type.
    /// - Returns: A new data frame.
    public func joined R  T  _        R
ColumnID T          ColumnID T          JoinKind
DataFrame where R   DataFrameProtocol  T   Hashable


/// A collection that represents a selection, potentially with gaps, of elements from
a typed column.
///
/// A column slice contains only certain elements from its parent column.
/// Create a slice by selecting certain elements.
/// For example, use ``filter(_:)`` to create a slice that only includes
elements with even values.
///
/// ```swift
/// let slice = column.filter({ $0.isMultiple(of: 2) })
/// ```
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public struct DiscontiguousColumnSlice WrappedElement
OptionalColumnProtocol
```

```
    /// The type of the column slice's elements.
    public typealias Element = WrappedElement

    /// The type that represents a position in the column slice.
    public typealias Index = Int

    /// The name of the slice's parent column.
    public var name: String

    /// The underlying type of the column's elements.
    @inlinable public var wrappedElementType: any Any
.Type { get }

    /// A prototype that creates type-erased columns with the same underlying
type as the column slice.
    ///
    /// Use a type-erased column prototype to create new columns of the same
type as the slice's parent column
    /// without explicitly knowing what type it is.
    public var prototype: any AnyColumnPrototype { get }

    /// Creates a slice with the contents of a column.
    ///
    /// - Parameter column: A column.
    public init(_ column: Column<WrappedElement>)

    /// Creates a slice with the contents of a column.
    ///
    /// - Parameter column: A column.
    /// - Parameter ranges: An array of integer ranges.
    public init(_ column: Column<WrappedElement>,
in ranges: [Range<Int>])

    /// Creates a new column by applying a transformation to each element.
    ///
    /// - Parameter transform: A closure that transforms the column
slice's elements to another type.
    public func map<T>(_
transform: (DiscontiguousColumnSlice<WrappedElement>.Element) throws ->
T) rethrows -> Column<T>

    /// Generates a slice that contains the elements that satisfy the predicate.
    ///
    /// - Parameter isIncluded: A predicate closure that returns a
Boolean.
    /// The method uses the closure to determine whether it includes an element
in the slice.
    public func filter(_
isIncluded: (DiscontiguousColumnSlice<WrappedElement>.Element) throws ->
Bool) rethrows -> DiscontiguousColumnSlice<WrappedElement>
```

```
    /// Generates a type-erased copy of the column slice.
    public func eraseToAnyColumn() -> AnyColumnSlice


@available(macOS 12, iOS 15, tvOS 15, watchOS 8, *)
extension DiscontiguousColumnSlice where WrappedElement:
Hashable {

    /// Generates a categorical summary of the column slice's elements.
    public func summary() ->
CategoricalSummary<WrappedElement>


@available(macOS 12, iOS 15, tvOS 15, watchOS 8, *)
extension DiscontiguousColumnSlice {

    /// Modifies a column slice by adding a value to each element.
    ///
    /// - Parameters:
    ///   - lhs: A column slice.
    ///   - rhs: A value of the same type as the column's elements.
    public static func += (lhs: inout
DiscontiguousColumnSlice<WrappedElement>, rhs: WrappedElement)
where WrappedElement: AdditiveArithmetic {

    /// Modifies a column slice by subtracting a value from each element.
    ///
    /// - Parameters:
    ///   - lhs: A column slice.
    ///   - rhs: A value of the same type as the column's elements.
    public static func -= (lhs: inout
DiscontiguousColumnSlice<WrappedElement>, rhs: WrappedElement)
where WrappedElement: AdditiveArithmetic {

    /// Modifies a column slice by multiplying each element by a value.
    ///
    /// - Parameters:
    ///   - lhs: A column slice.
    ///   - rhs: A value of the same type as the column's elements.
    public static func *= (lhs: inout
DiscontiguousColumnSlice<WrappedElement>, rhs: WrappedElement)
where WrappedElement: Numeric {

    /// Modifies an integer column slice by dividing each element by a value.
    ///
    /// - Parameters:
    ///   - lhs: A column slice.
    ///   - rhs: A value of the same type as the column's elements.
```

```
    public static func          inout
DiscontiguousColumnSlice WrappedElement        WrappedElement
where WrappedElement   BinaryInteger

    /// Modifies a floating-point column slice by dividing each element by a value.
    ///
    /// - Parameters:
    ///   - lhs: A column slice.
    ///   - rhs: A value of the same type as the column's elements.
    public static func          inout
DiscontiguousColumnSlice WrappedElement        WrappedElement
where WrappedElement   FloatingPoint


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DiscontiguousColumnSlice   Sendable where
WrappedElement   Sendable


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DiscontiguousColumnSlice   BidirectionalCollection
MutableCollection

    /// The index of the initial element in the column slice.
    public var startIndex  Int   get

    /// The index of the final element in the column slice.
    public var endIndex  Int   get

    /// Returns the index immediately after an element index.
    /// - Parameter i: A valid index to an element in the column slice.
    public func index          Int     Int

    /// Returns the index immediately before an element index.
    /// - Parameter i: A valid index to an element in the column slice.
    public func index          Int     Int

    /// The number of elements in the column slice.
    public var count  Int   get

    /// The number of missing elements in the column slice.
    @available macOS 12  iOS 15  tvOS 15  watchOS 8
    public var missingCount  Int   get

    /// Accesses an element at an index.
    ///
    /// - Parameter position: A valid index to an element in the column
slice.
    public subscript          Int
```

```
DiscontiguousColumnSlice WrappedElement  Element

    /// Returns a Boolean that indicates whether the element at the index is
missing.
    ///
    /// - Parameter index: An index.
    public func isNil           Int     Bool

    /// Accesses a contiguous range of elements.
    ///
    /// - Parameter range: A range of valid indices in the column slice.
    public subscript        Range Int
DiscontiguousColumnSlice WrappedElement

    /// Accesses a contiguous range of elements with a range expression.
    ///
    /// - Parameter range: A range expression of valid indices in the
column slice.
    @inlinable public subscript R         R
DiscontiguousColumnSlice WrappedElement  where R
RangeExpression  R Bound    Int

    /// Accesses a contiguous range of elements with an unbounded range.
    ///
    /// - Parameter range: An unbounded range of valid indices in the
column slice.
    @inlinable public subscript         UnboundedRange_
       DiscontiguousColumnSlice WrappedElement

    /// A type that represents the indices that are valid for subscripting the
    /// collection, in ascending order.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias Indices
DefaultIndices DiscontiguousColumnSlice WrappedElement

    /// A type that provides the collection's iteration interface and
    /// encapsulates its iteration state.
    ///
    /// By default, a collection conforms to the `Sequence` protocol by
    /// supplying `IndexingIterator` as its associated `Iterator`
    /// type.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias Iterator
IndexingIterator DiscontiguousColumnSlice WrappedElement

    /// A collection representing a contiguous subrange of this collection's
    /// elements. The subsequence shares indices with the original collection.
    ///
    /// The default subsequence type for collections that don't define their own
    /// is `Slice`.
```

```
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias SubSequence
DiscontiguousColumnSlice WrappedElement


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DiscontiguousColumnSlice   Equatable where
WrappedElement   Equatable

    /// Returns a Boolean that indicates whether the column slices are equal.
    /// - Parameters:
    ///   - lhs: A discontiguous column slice.
    ///   - rhs: Another discontiguous column slice.
    public static func
DiscontiguousColumnSlice WrappedElement
DiscontiguousColumnSlice WrappedElement      Bool


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DiscontiguousColumnSlice   Hashable where
WrappedElement   Hashable

    /// Hashes the essential components of the column slice by feeding them
into a hasher.
    /// - Parameter hasher: A hasher the method uses to combine the
components of the column slice.
    public func hash              inout Hasher

    /// Generates a discontiguous slice that contains unique elements.
    ///
    /// The method only adds the first of multiple elements with the same value
    /// --- the element with the smallest index ---
    /// to the slice.
    ///
    /// - Returns: A discontiguous column slice.
    public func distinct
DiscontiguousColumnSlice WrappedElement

    /// The hash value.
    ///
    /// Hash values are not guaranteed to be equal across different executions of
    /// your program. Do not save hash values to use during a future execution.
    ///
    /// - Important: `hashValue` is deprecated as a `Hashable`
requirement. To
    ///   conform to `Hashable`, implement the `hash(into:)` requirement
instead.
    ///   The compiler provides an implementation for `hashValue` for you.
    public var hashValue  Int   get
```

```swift
@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DiscontiguousColumnSlice

    /// Modifies a column slice by adding each value in a collection to
    /// the corresponding element in the column.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the
column's elements.
    public static func   C      inout
DiscontiguousColumnSlice WrappedElement       C  where
WrappedElement   AdditiveArithmetic  WrappedElement
C Element  C   Collection

    /// Modifies a column slice by adding each optional value in a collection to
    /// the corresponding element in the column.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the
column's elements.
    public static func   C      inout
DiscontiguousColumnSlice WrappedElement       C  where
WrappedElement   AdditiveArithmetic  C   Collection  C Element
   WrappedElement

    /// Modifies a column slice by subtracting each value in a collection from
    /// the corresponding element in the column.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the
column's elements.
    public static func   C      inout
DiscontiguousColumnSlice WrappedElement       C  where
WrappedElement   AdditiveArithmetic  WrappedElement
C Element  C   Collection

    /// Modifies a column slice by subtracting each optional value in a collection
from
    /// the corresponding element in the column.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the
column's elements.
    public static func   C      inout
DiscontiguousColumnSlice WrappedElement       C  where
```

```
WrappedElement  AdditiveArithmetic  C  Collection  C Element
   WrappedElement

    /// Modifies a column slice by multiplying each element in the column by
    /// the corresponding value in a collection.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the
column's elements.
    public static func    C       inout
DiscontiguousColumnSlice WrappedElement       C  where
WrappedElement  Numeric  WrappedElement   C Element  C
Collection

    /// Modifies a column slice by multiplying each element in the column by
    /// the corresponding optional value in a collection.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the
column's elements.
    public static func    C       inout
DiscontiguousColumnSlice WrappedElement       C  where
WrappedElement  Numeric  C  Collection  C Element
WrappedElement

    /// Modifies an integer column slice by dividing each element in the column
by
    /// the corresponding value in a collection.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the
column's elements.
    public static func    C       inout
DiscontiguousColumnSlice WrappedElement       C  where
WrappedElement  BinaryInteger  WrappedElement   C Element  C
  Collection

    /// Modifies an integer column slice by dividing each element in the column
by
    /// the corresponding optional value in a collection.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the
column's elements.
    public static func    C       inout
DiscontiguousColumnSlice WrappedElement       C  where
```

```
WrappedElement   BinaryInteger  C   Collection  C Element
WrappedElement

    /// Modifies a floating-point column slice by dividing each element in the
column by
    /// the corresponding value in a collection.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the
column's elements.
    public static func     C       inout
DiscontiguousColumnSlice WrappedElement        C  where
WrappedElement   FloatingPoint  WrappedElement    C Element  C
  Collection

    /// Modifies a floating-point column slice by dividing each element in the
column by
    /// the corresponding optional value in a collection.
    ///
    /// - Parameters:
    ///   - lhs: A column.
    ///   - rhs: A collection that contains elements of the same type as the
column's elements.
    public static func     C       inout
DiscontiguousColumnSlice WrappedElement        C  where
WrappedElement   FloatingPoint  C   Collection  C Element
WrappedElement


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DiscontiguousColumnSlice where WrappedElement
Comparable

    /// Returns the element with the lowest value, ignoring missing elements.
    public func min
DiscontiguousColumnSlice WrappedElement  Element

    /// Returns the element with the highest value, ignoring missing elements.
    public func max
DiscontiguousColumnSlice WrappedElement  Element

    /// Returns the index of the element with the lowest value, ignoring missing
elements.
    public func argmin      Int

    /// Returns the index of the element with the highest value, ignoring missing
elements.
    public func argmax      Int
```

```
@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DiscontiguousColumnSlice where WrappedElement
AdditiveArithmetic

    /// Returns the sum of the column slice's elements, ignoring missing
elements.
    public func sum      WrappedElement


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DiscontiguousColumnSlice where WrappedElement
FloatingPoint

    /// Returns the mean average of the floating-point slice's elements, ignoring
missing elements.
    public func mean
DiscontiguousColumnSlice WrappedElement  Element

    /// Returns the standard deviation of the floating-point column slice's
elements, ignoring missing elements.
    ///
    /// - Parameter deltaDegreesOfFreedom: A nonnegative integer.
    /// The method calculates the standard deviation's divisor by subtracting this
parameter from the number of
    /// non-`nil` elements (`n - deltaDegreesOfFreedom` where `n` is
the number of non-`nil` elements).
    ///
    /// - Returns: The standard deviation; otherwise, `nil` if there are
fewer than
    /// `deltaDegreesOfFreedom + 1` non-`nil` items in the column.
    public func standardDeviation                          Int
1     DiscontiguousColumnSlice WrappedElement  Element


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DiscontiguousColumnSlice where WrappedElement
BinaryInteger

    /// Returns the mean average of the integer slice's elements, ignoring
missing elements.
    public func mean      Double

    /// Returns the standard deviation of the integer column slice's elements,
ignoring missing elements.
    ///
    /// - Parameter deltaDegreesOfFreedom: A nonnegative integer.
    /// The method calculates the standard deviation's divisor by subtracting this
parameter from the number of
    /// non-`nil` elements (`n - deltaDegreesOfFreedom` where `n` is
```

```
the number of non-`nil` elements).
    ///
    /// - Returns: The standard deviation; otherwise, `nil` if there are
fewer than
    /// `deltaDegreesOfFreedom + 1` non-`nil` items in the column.
    public func standardDeviation                          Int
1     Double


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DiscontiguousColumnSlice   CustomStringConvertible
CustomDebugStringConvertible  CustomReflectable

    /// A text representation of the column slice.
    public var description  String   get

    /// A text representation of the column slice suitable for debugging.
    public var debugDescription  String   get

    /// A mirror that reflects the column slice.
    public var customMirror  Mirror   get


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DiscontiguousColumnSlice where WrappedElement
BinaryFloatingPoint

    /// Generates a numeric summary of the floating-point column slice's
elements.
    public func numericSummary
NumericSummary WrappedElement


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension DiscontiguousColumnSlice where WrappedElement
BinaryInteger

    /// Generates a numeric summary of the integer column slice's elements.
    public func numericSummary      NumericSummary Double


/// A view on a column that replaces missing elements with a default value.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public struct FilledColumn Base    ColumnProtocol where Base
OptionalColumnProtocol

    /// The type of the column's elements that defines an associated type for the
bidirectional collection protocol.
    ///
    /// See
```

<doc://com.apple.documentation/documentation/Swift/BidirectionalCollection> for more information.

```
    public typealias Element   Base WrappedElement

    /// The type of the column's elements that defines an associated type for the optional column protocol.
    ///
    /// See ``OptionalColumnProtocol`` for more information.
    public typealias WrappedElement   Base WrappedElement

    /// The name of the column.
    public var name  String

    /// The index of the initial element in the column.
    @inlinable public var startIndex  Base Index   get

    /// The index of the final element in the column.
    @inlinable public var endIndex  Base Index   get

    /// Returns the position immediately after an index.
    /// - Parameter i: A valid index to a row in the grouping.
    @inlinable public func index          Base Index
Base Index

    /// Returns the row index immediately before a row index.
    /// - Parameter i: A valid index to a row in the grouping.
    @inlinable public func index           Base Index
Base Index

    /// Retrieves an element at a position in the column type.
    ///
    /// - Parameter position: A valid index in the column type.
    @inlinable public subscript           Base Index
Base WrappedElement   get

    /// A type that represents a position in the collection.
    ///
    /// Valid indices consist of the position of every element and a
    /// "past the end" position that's not valid for use as a subscript
    /// argument.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias Index   Base Index

    /// A type that represents the indices that are valid for subscripting the
    /// collection, in ascending order.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias Indices
DefaultIndices FilledColumn Base

    /// A type that provides the collection's iteration interface and
```

```
    /// encapsulates its iteration state.
    ///
    /// By default, a collection conforms to the `Sequence` protocol by
    /// supplying `IndexingIterator` as its associated `Iterator`
    /// type.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias Iterator
IndexingIterator FilledColumn Base

    /// A collection representing a contiguous subrange of this collection's
    /// elements. The subsequence shares indices with the original collection.
    ///
    /// The default subsequence type for collections that don't define their own
    /// is `Slice`.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias SubSequence   Slice FilledColumn Base


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension FilledColumn where Base WrappedElement   Hashable

    /// Generates a categorical summary of the filled column's elements,
including default values.
    public func summary
CategoricalSummary FilledColumn Base  WrappedElement


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension FilledColumn where Base WrappedElement   Comparable


    /// Returns the element with the lowest value.
    public func min      FilledColumn Base  Element

    /// Returns the element with the highest value.
    public func max      FilledColumn Base  Element

    /// Returns the index of the element with the lowest value.
    public func argmin      Base Index

    /// Returns the index of the element with the highest value.
    public func argmax      FilledColumn Base  Index


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension FilledColumn where Base WrappedElement
BinaryInteger

    /// Returns the sum of the integer column's elements.
```

```
    public func sum      FilledColumn Base  Element

    /// Returns the mean average of the integer column's elements.
    public func mean      Double

    /// Returns the standard deviation of the integer column's elements.
    ///
    /// - Parameter deltaDegreesOfFreedom: A nonnegative integer.
    /// The method calculates the standard deviation's divisor by subtracting this
parameter from the number of
    /// non-`nil` elements (`n - deltaDegreesOfFreedom` where `n` is
the number of non-`nil` elements).
    ///
    /// - Returns: The standard deviation; otherwise, `nil` if there are
fewer than
    /// `deltaDegreesOfFreedom + 1` non-`nil` items in the column.
    public func standardDeviation                          Int
1     Double


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension FilledColumn where Base WrappedElement
FloatingPoint

    /// Returns the sum of the floating-point column's elements.
    public func sum      FilledColumn Base  Element

    /// Returns the mean average of the floating-point column's elements.
    public func mean      FilledColumn Base  Element

    /// Returns the standard deviation of the floating-point column's elements.
    ///
    /// - Parameter deltaDegreesOfFreedom: A nonnegative integer.
    /// The method calculates the standard deviation's divisor by subtracting this
parameter from the number of
    /// non-`nil` elements (`n - deltaDegreesOfFreedom` where `n` is
the number of non-`nil` elements).
    ///
    /// - Returns: The standard deviation; otherwise, `nil` if there are
fewer than
    /// `deltaDegreesOfFreedom + 1` non-`nil` items in the column.
    public func standardDeviation                          Int
1     FilledColumn Base  Element


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension FilledColumn   Sendable where Base   Sendable
Base WrappedElement   Sendable
```

```
@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension FilledColumn   CustomStringConvertible

    /// A mirror that reflects the filled column.
    public var description  String   get

    /// A text representation of the filled column suitable for debugging.
    public var debugDescription  String   get

    /// Generates a string description of the filled column.
    ///
    /// - Parameter options: The formatting options.
    public func description          FormattingOptions
String


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension FilledColumn where Base WrappedElement
BinaryFloatingPoint

    /// Generates a numeric summary of the floating-point column's elements.
    public func numericSummary
NumericSummary Base WrappedElement


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension FilledColumn where Base WrappedElement
BinaryInteger

    /// Generates a numeric summary of the integer column's elements.
    public func numericSummary      NumericSummary Double


/// A set of parameters that indicate how to present the contents of data frame or
column types to a printable string.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public struct FormattingOptions

    /// The largest number of characters a description can generate per line.
    public var maximumLineWidth  Int

    /// The largest number of characters a description can generate per cell.
    public var maximumCellWidth  Int

    /// The largest number of rows a description can generate.
    public var maximumRowCount  Int

    /// A Boolean value that indicates whether the description includes the
column types.
    public var includesColumnTypes  Bool
```

```
    /// A Boolean value that indicates whether the description includes the row indices.
    @available macOS 14  iOS 17  tvOS 17  watchOS 10
    public var includesRowIndices  Bool

    /// A Boolean value that indicates whether the description includes the number of rows and columns.
    @available macOS 14  iOS 17  tvOS 17  watchOS 10
    public var includesRowAndColumnCounts  Bool

    /// The floating point format style.
    @available macOS 12.3  iOS 15.4  tvOS 15.4  watchOS 8.5

    public var floatingPointFormatStyle FloatingPointFormatStyle Double

    /// The integer format style.
    @available macOS 12.3  iOS 15.4  tvOS 15.4  watchOS 8.5

    public var integerFormatStyle  IntegerFormatStyle Int

    /// The date format style.
    @available macOS 12.3  iOS 15.4  tvOS 15.4  watchOS 8.5

    public var dateFormatStyle  Date FormatStyle

    /// The locale.
    @available macOS 12.3  iOS 15.4  tvOS 15.4  watchOS 8.5

    public var locale  Locale

    /// Creates a formatting options instance with default parameters.
    public init

    /// Creates a formatting options instance with a locale.
    @available macOS 12.3  iOS 15.4  tvOS 15.4  watchOS 8.5

    public init          Locale

    /// Creates a formatting options instance.
    /// - Parameters:
    ///    - maximumLineWidth: The largest number of characters a description can generate per line.
    ///    - maximumCellWidth: The largest number of characters a description can generate per cell.
    ///    - maximumRowCount: The largest number of rows a description can generate.
    ///    - includesColumnTypes: A Boolean that indicates whether the description prints a column's type.
```

```
    public init                    Int                    Int
50                  Int   20                    Bool
true


/// A collection of group summaries.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public protocol GroupSummaries   CustomStringConvertible

    /// Retrieves one or more summaries by their group keys.
    ///
    /// - Parameter keys: A comma-separated, or variadic, list of key
optionals.
    subscript       Any         DataFrame    get

    /// A text representation of the group summaries.
    override var description  String   get

    /// Generates a text representation of the group summaries.
    ///
    /// - Parameter options: A ``FormattingOptions`` instance.
    ///
    /// See ``DataFrame/description(options:)`` for more information.
    func description          FormattingOptions     String


/// A JSON reading error.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public enum JSONReadingError   Error

    /// An error that occurs when the JSON structure is incompatible with a data
frame.
    case unsupportedStructure

    /// An error that occurs when the JSON data contains a value of the wrong
type for a type-constrained column.
    ///
    /// - Parameters:
    ///   - row: The index of the row that contains the incorrect value.
    ///   - column: The name of the column that contains the incorrect value.
    ///   - expectedType: The expected type.
    ///   - value: The JSON value.
    case wrongType      Int          String
JSONType         any Sendable

    /// An error that occurs when the JSON data contains incompatible values in
a column.
    ///
    /// - Parameters:
    ///   - column: The name of the column that contains the incompatible
```

```
values.
    case incompatibleValues         String

    /// An error that occurs when a JSON value fails to parse as the specified
type.
    ///
    /// - Parameters:
    ///   - row: The index of the row that contains the incorrect value.
    ///   - column: The name of the column that contains the incorrect value.
    ///   - expectedType: The expected type.
    ///   - value: The JSON value.
    case failedToParse      Int          String
JSONType            String


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension JSONReadingError   CustomStringConvertible

    /// A text representation of the error.
    public var description  String   get


@available macOS 14  iOS 17  tvOS 17  watchOS 10
extension JSONReadingError   LocalizedError

    /// A localized message describing what error occurred.
    public var errorDescription  String    get


/// A set of JSON file-reading options.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public struct JSONReadingOptions

    /// An array of closures that parse a date from a string.
    public var dateParsers    String     Date

    /// Creates a set of options for reading a JSON file.
    public init

    /// Adds a date parsing strategy.
    /// - Parameter strategy: A parsing strategy that has a string input
and a date output.
    public mutating func addDateParseStrategy T  _
T  where T   ParseStrategy  T ParseInput    String
T ParseOutput    Date


/// Represents the value types in a JSON file.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public enum JSONType   Sendable
```

```
    /// An integer type.
    case integer

    /// A Boolean type.
    case boolean

    /// A double-precision floating-point type.
    case double

    /// A date type.
    case date

    /// A string type.
    case string

    /// An array type.
    case array

    /// An object type.
    case object

    /// Returns a Boolean value indicating whether two values are equal.
    ///
    /// Equality is the inverse of inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is `false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to compare.
    public static func        JSONType     JSONType     Bool

    /// Hashes the essential components of this value by feeding them into the
    /// given hasher.
    ///
    /// Implement this method to conform to the `Hashable` protocol. The
    /// components used for hashing must be the same as the components
compared
    /// in your type's `==` operator implementation. Call
`hasher.combine(_:)`
    /// with each of these components.
    ///
    /// - Important: In your implementation of `hash(into:)`,
    ///   don't call `finalize()` on the `hasher` instance provided,
    ///   or replace it with a different instance.
    ///   Doing so may become a compile-time error in the future.
    ///
    /// - Parameter hasher: The hasher to use when combining the
components
    ///   of this instance.
```

```
    public func hash                inout Hasher

    /// The hash value.
    ///
    /// Hash values are not guaranteed to be equal across different executions of
    /// your program. Do not save hash values to use during a future execution.
    ///
    /// - Important: `hashValue` is deprecated as a `Hashable`
requirement. To
    ///     conform to `Hashable`, implement the `hash(into:)` requirement
instead.
    ///     The compiler provides an implementation for `hashValue` for you.
    public var hashValue  Int   get


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension JSONType   Equatable


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension JSONType   Hashable


/// A set of JSON file-reading options.
@available macOS 13  iOS 16  tvOS 16  watchOS 9
public struct JSONWritingOptions

    /// A Boolean value that indicates whether to sort the keys.
    ///
    /// Defaults to `false`.
    public var sortKeys  Bool

    /// A Boolean value that indicates whether to split lines and indent the
generated JSON.
    ///
    /// Defaults to `false`.
    public var prettyPrint  Bool

    /// A closure that maps dates to their string representations.
    ///
    /// Defaults to ISO 8601 encoding.
    public var dateFormatter   Date     String

    /// Creates a set of options for writing a JSON file.
    public init


/// An operation type that joins two data frame types.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public enum JoinKind   Sendable
```

```
    /// A join kind that only contains rows with matching values in both data
frame types.
    case inner

    /// A join kind that contains all rows from the left data frame type,
    /// and only the rows with matching values from the right data frame type.
    case left

    /// A join kind that contains all rows from the right data frame type,
    /// and only the rows with matching values from the left data frame type.
    case right

    /// A join kind that contains every row from both data frame types.
    case full

    /// Returns a Boolean value indicating whether two values are equal.
    ///
    /// Equality is the inverse of inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is `false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to compare.
    public static func        JoinKind     JoinKind     Bool

    /// Hashes the essential components of this value by feeding them into the
    /// given hasher.
    ///
    /// Implement this method to conform to the `Hashable` protocol. The
    /// components used for hashing must be the same as the components
compared
    /// in your type's `==` operator implementation. Call
`hasher.combine(_:)`
    /// with each of these components.
    ///
    /// - Important: In your implementation of `hash(into:)`,
    ///   don't call `finalize()` on the `hasher` instance provided,
    ///   or replace it with a different instance.
    ///   Doing so may become a compile-time error in the future.
    ///
    /// - Parameter hasher: The hasher to use when combining the
components
    ///   of this instance.
    public func hash              inout Hasher

    /// The hash value.
    ///
    /// Hash values are not guaranteed to be equal across different executions of
    /// your program. Do not save hash values to use during a future execution.
```

```
    ///
    /// - Important: `hashValue` is deprecated as a `Hashable`
requirement. To
    ///     conform to `Hashable`, implement the `hash(into:)` requirement
instead.
    ///     The compiler provides an implementation for `hashValue` for you.
    public var hashValue  Int   get


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension JoinKind   Equatable


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension JoinKind   Hashable


/// A summary of a numerical column.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public struct NumericSummary Element    Hashable
CustomDebugStringConvertible where Element
BinaryFloatingPoint

    /// The number of non-missing elements in the column.
    public var someCount  Int

    /// The number of missing elements in the column.
    public var noneCount  Int

    /// The total number of elements in the column.
    public var totalCount  Int   get

    /// The midpoint value that's above half of the non-missing elements' values
and below the other half's values.
    public var median  Element

    /// The value that's above 25% of the non-missing elements' values.
    public var firstQuartile  Element

    /// The value that's above 75% of the non-missing elements' values.
    public var thirdQuartile  Element

    /// The arithmetic mean of a column's values that excludes missing
elements.
    public var mean  Element

    /// The standard deviation of a column's values that excludes missing
elements.
    public var standardDeviation  Element
```

```
    /// The smallest value, excluding missing elements.
    public var min  Element

    /// The largest value, excluding missing elements.
    public var max  Element

    /// Creates an empty numeric summary with default values.
    public init

    /// Creates an empty numeric summary.
    ///
    /// - Parameters:
    ///   - someCount: The number of elements in column, excluding
missing elements.
    ///   - noneCount: The number of missing elements in the column.
    ///   - mean: The arithmetic mean of a column's values, excluding
missing elements.
    ///   - standardDeviation: The standard deviation of a column's
values, excluding missing elements.
    ///   - min: The smallest value, excluding missing elements.
    ///   - max: The largest value, excluding missing elements.
    ///   - median: The midpoint value that bisects the values of the non-
missing elements.
    ///   - firstQuartile: The value that's above 25% of the values non-
missing elements' values.
    ///   - thirdQuartile: The value that's above 75% of the non-missing
elements' values.
    public init             Int             Int        Element
                    Element       Element       Element
        Element                 Element
Element

    /// A text representation of the summary's statistics suitable for debugging.
    public var debugDescription  String   get

    /// Hashes the essential components of this value by feeding them into the
    /// given hasher.
    ///
    /// Implement this method to conform to the `Hashable` protocol. The
    /// components used for hashing must be the same as the components
compared
    /// in your type's `==` operator implementation. Call
`hasher.combine(_:)`
    /// with each of these components.
    ///
    /// - Important: In your implementation of `hash(into:)`,
    ///   don't call `finalize()` on the `hasher` instance provided,
    ///   or replace it with a different instance.
    ///   Doing so may become a compile-time error in the future.
    ///
```

```
    /// - Parameter hasher: The hasher to use when combining the components
    ///   of this instance.
    public func hash                inout Hasher

    /// Returns a Boolean value indicating whether two values are equal.
    ///
    /// Equality is the inverse of inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is `false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to compare.
    public static func        NumericSummary Element
NumericSummary Element      Bool

    /// The hash value.
    ///
    /// Hash values are not guaranteed to be equal across different executions of
    /// your program. Do not save hash values to use during a future execution.
    ///
    /// - Important: `hashValue` is deprecated as a `Hashable` requirement. To
    ///   conform to `Hashable`, implement the `hash(into:)` requirement instead.
    ///   The compiler provides an implementation for `hashValue` for you.
    public var hashValue  Int   get


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension NumericSummary   Sendable where Element   Sendable


/// A type that represents a column that has missing values.
///
/// `OptionalColumnProtocol` defines the common functionality for column types that support missing values.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public protocol OptionalColumnProtocol
ColumnProtocol

    /// The type of the optional column type's elements.
    associatedtype WrappedElement where Self Element
Self WrappedElement


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension OptionalColumnProtocol

    /// Generates a filled column by replacing missing elements with a value.
```

```
    ///
    /// - Parameter value: A value the method uses to replace the
column's missing elements.
    /// - Returns: A filled column.
    public func filled              Self WrappedElement
FilledColumn Self


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension OptionalColumnProtocol

    /// Generates a string description of the optional column type.
    ///
    /// - Parameter options: The formatting options.
    public func description          FormattingOptions
String


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension OptionalColumnProtocol where Self WrappedElement
AdditiveArithmetic

    /// Generates a column by adding each element in an optional column type
to the corresponding elements of another.
    /// - Parameters:
    ///   - lhs: An optional column type.
    ///   - rhs: Another optional column type.
    /// - Returns: A new column.
    public static func          Self       Self
Column Self WrappedElement

    /// Generates a column by subtracting each element in an optional column
type from
    /// the corresponding elements of another.
    /// - Parameters:
    ///   - lhs: An optional column type.
    ///   - rhs: Another optional column type.
    /// - Returns: A new column.
    public static func          Self       Self
Column Self WrappedElement


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension OptionalColumnProtocol where Self WrappedElement
Numeric

    /// Generates a column by multiplying each element in an optional column
type
    /// by the corresponding elements of another.
    /// - Parameters:
```

```
    ///     - lhs: An optional column type.
    ///     - rhs: Another optional column type.
    /// - Returns: A new column.
    public static func         Self       Self
Column Self WrappedElement


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension OptionalColumnProtocol where Self WrappedElement
BinaryInteger

    /// Generates an integer column by dividing each element in an optional
column type
    /// by the corresponding elements of another.
    /// - Parameters:
    ///     - lhs: An optional column type.
    ///     - rhs: Another optional column type.
    /// - Returns: A new column.
    public static func         Self       Self
Column Self WrappedElement


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension OptionalColumnProtocol where Self WrappedElement
FloatingPoint

    /// Generates a floating-point column by dividing each element in an optional
column type
    /// by the corresponding elements of another.
    /// - Parameters:
    ///     - lhs: An optional column type.
    ///     - rhs: Another optional column type.
    /// - Returns: A new column.
    public static func         Self       Self
Column Self WrappedElement


extension OptionalColumnProtocol

    /// Generates a column by adding a value to each element in an optional
column.
    /// - Parameters:
    ///     - lhs: An optional column type.
    ///     - rhs: A value of the same type as the optional column.
    /// - Returns: A new column.
    @available macOS 12  iOS 15  tvOS 15  watchOS 8
    public static func         Self       Self WrappedElement
   Column Self WrappedElement  where Self WrappedElement
AdditiveArithmetic
```

```
    /// Generates a column by adding each element in an optional column to a value.
    /// - Parameters:
    ///   - lhs: A value of the same type as the optional column's type.
    ///   - rhs: An optional column type.
    /// - Returns: A new column.
    @available macOS 12  iOS 15  tvOS 15  watchOS 8
    public static func         Self WrappedElement       Self
   Column Self WrappedElement  where Self WrappedElement
AdditiveArithmetic

    /// Generates a column by subtracting a value from each element in an optional column type.
    /// - Parameters:
    ///   - lhs: An optional column type.
    ///   - rhs: A value of the same type as the optional column's type.
    /// - Returns: A new column.
    @available macOS 12  iOS 15  tvOS 15  watchOS 8
    public static func         Self       Self WrappedElement
   Column Self WrappedElement  where Self WrappedElement
AdditiveArithmetic

    /// Generates a column by subtracting each element in an optional column from a value.
    /// - Parameters:
    ///   - lhs: A value of the same type as the optional column's type.
    ///   - rhs: An optional column type.
    /// - Returns: A new column.
    @available macOS 12  iOS 15  tvOS 15  watchOS 8
    public static func         Self WrappedElement       Self
   Column Self WrappedElement  where Self WrappedElement
AdditiveArithmetic


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension OptionalColumnProtocol where Self WrappedElement
Numeric

    /// Generates a column by multiplying each element in an optional column by a value.
    /// - Parameters:
    ///   - lhs: An optional column type.
    ///   - rhs: A value of the same type as the optional column's type.
    /// - Returns: A new column.
    public static func         Self       Self WrappedElement
   Column Self WrappedElement

    /// Generates a column by multiplying a value by each element in an optional column type.
    /// - Parameters:
```

```
    ///   - lhs: A value of the same type as the optional column's type.
    ///   - rhs: An optional column type.
    /// - Returns: A new column.
    public static func / (Self.WrappedElement, Self) -> Column<Self.WrappedElement>
```

```
@available(macOS 12, iOS 15, tvOS 15, watchOS 8)
extension OptionalColumnProtocol where Self.WrappedElement: BinaryInteger
```

```
    /// Generates an integer column by dividing each element in an optional column by a value.
    /// - Parameters:
    ///   - lhs: An optional column type.
    ///   - rhs: A value of the same type as the optional column's type.
    /// - Returns: A new column.
    public static func / (Self, Self.WrappedElement) -> Column<Self.WrappedElement>
```

```
    /// Generates an integer column by dividing a value by each element in an optional column type.
    /// - Parameters:
    ///   - lhs: A value of the same type as the optional column's type.
    ///   - rhs: An optional column type.
    /// - Returns: A new column.
    public static func / (Self.WrappedElement, Self) -> Column<Self.WrappedElement>
```

```
@available(macOS 12, iOS 15, tvOS 15, watchOS 8)
extension OptionalColumnProtocol where Self.WrappedElement: FloatingPoint
```

```
    /// Generates a floating-point column by dividing each element in an optional column by a value.
    /// - Parameters:
    ///   - lhs: An optional column type.
    ///   - rhs: A value of the same type as the optional column's type.
    /// - Returns: A new column.
    public static func / (Self, Self.WrappedElement) -> Column<Self.WrappedElement>
```

```
    /// Generates a floating-point column by dividing a value by each element in an optional column type.
    /// - Parameters:
    ///   - lhs: A value of the same type as the optional column.
    ///   - rhs: An optional column type.
    /// - Returns: A new column.
    public static func / (Self.WrappedElement, Self) -> Column<Self.WrappedElement>
```

```
    Column Self WrappedElement


/// A type that represents a sort ordering.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public enum Order   Sendable

    /// A sort ordering that starts with the lowest value and monotonically
proceeds to higher values.
    case ascending

    /// A sort ordering that starts with the highest value and monotonically
proceeds to lower values.
    case descending

    /// Returns a Boolean that indicates whether the comparable types match the
order's state.
    ///
    /// - Parameters:
    ///   - lhs: A comparable type.
    ///   - rhs: Another comparable type.
    public func areOrdered T  _      T  _      T     Bool
where T   Comparable

    /// Returns a Boolean value indicating whether two values are equal.
    ///
    /// Equality is the inverse of inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is `false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to compare.
    public static func        Order     Order     Bool

    /// Hashes the essential components of this value by feeding them into the
    /// given hasher.
    ///
    /// Implement this method to conform to the `Hashable` protocol. The
    /// components used for hashing must be the same as the components
compared
    /// in your type's `==` operator implementation. Call
`hasher.combine(_:)`
    /// with each of these components.
    ///
    /// - Important: In your implementation of `hash(into:)`,
    ///   don't call `finalize()` on the `hasher` instance provided,
    ///   or replace it with a different instance.
    ///   Doing so may become a compile-time error in the future.
    ///
    /// - Parameter hasher: The hasher to use when combining the
```

```
components
    ///     of this instance.
    public func hash                inout Hasher

    /// The hash value.
    ///
    /// Hash values are not guaranteed to be equal across different executions of
    /// your program. Do not save hash values to use during a future execution.
    ///
    /// - Important: `hashValue` is deprecated as a `Hashable`
requirement. To
    ///     conform to `Hashable`, implement the `hash(into:)` requirement
instead.
    ///     The compiler provides an implementation for `hashValue` for you.
    public var hashValue  Int   get


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension Order   Equatable


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension Order   Hashable


/// A collection of row selections that have the same value in a column.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public struct RowGrouping GroupingKey    RowGroupingProtocol
where GroupingKey   Hashable

    /// A text representation of the row grouping.
    public var description  String   get

    /// Creates a row grouping from a list of groups.
    ///
    /// The member data frames must all have the same columns (count, names,
and types).
    ///
    /// - Parameters:
    ///   - groups: An array of tuples. Each tuple pairs a key with a data
frame type.
    ///   - groupKeysColumnName: The name of the grouping key column
the row grouping creates when it generates a data
    ///     frame, such as its ``ungrouped()`` or ``counts(order:)``
methods.
    public init D            GroupingKey   D
                        String  where D   DataFrameProtocol

    /// Generates a data frame with two columns, one that has a row for each
group key and another for the number of
```

```
    /// rows in the group.
    ///
    /// - Parameter order: A sorting order the method uses to sort the data frame by its count column.
    ///
    /// The name of the data frame's column that stores the number of rows in each group is *count*.
    public func counts(order: Order = .forward) -> DataFrame

    /// Generates a data frame by aggregating each group's contents for each column you list by name.
    ///
    /// - Parameters:
    ///   - columnNames: A comma-separated, or variadic, list of column names.
    ///   - naming: A closure that converts a column name to another name.
    ///   - transform: A closure that aggregates a group's elements in a specific column.
    ///
    /// The data frame contains two columns that:
    /// - Identify each group
    /// - Store the results of your aggregation transform closure
    public func aggregated<Element, Result>(
        _ columnNames: String...,
        naming: (String) -> String = { $0 },
        transform: (DiscontiguousColumnSlice<Element>) throws -> Result
    ) rethrows -> DataFrame

    /// Generates a data frame that contains all the rows from each group in the row grouping.
    ///
    /// The method discards any column with the same name as the row grouping itself.
    public func ungrouped() -> DataFrame

    /// Returns a row grouping containing only the groups that satisfy a predicate.
    ///
    /// - Parameter isIncluded: A predicate closure that takes a group and returns a Boolean that indicates whether
    ///   the group is included.
    /// - Returns: A data frame slice that contains the rows that satisfy the predicate.
    @available(macOS 12.3, iOS 15.4, tvOS 15.4, watchOS 8.5, *)
    public func filter(_ isIncluded: (DataFrame.Slice) throws -> Bool) rethrows -> RowGrouping<GroupingKey>

    /// Generates a row grouping that applies a transformation closure to each group in the original.
    ///
    /// - Parameter transform: A closure that generates a data frame from
```

```
a data frame slice that represents a group.
    public func mapGroups _              DataFrame Slice
throws    DataFrame  rethrows    RowGrouping GroupingKey

    /// Retrieves a group slice by key.
    ///
    /// - Parameter keys: A comma-separated, or variadic, list of key
optionals.
    public subscript       Any         DataFrame Slice
get


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension RowGrouping

    /// Generates two row groupings by randomly splitting the original with a
proportion and a seed number.
    /// - Parameters:
    ///   - proportion: A proportion in the range `[0.0, 1.0]`.
    ///   - seed: A seed number for a random-number generator.
    /// - Returns: A tuple of two row groupings.
    public func randomSplit                  Double        Int
  nil      RowGrouping GroupingKey   RowGrouping GroupingKey


/// Date based grouping
@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension RowGrouping

    /// Creates a row grouping from a column with date or time elements.
    /// - Parameters:
    ///   - frame: A data frame type.
    ///   - columnName: The name of the column that stores a row's date
and time information.
    ///   - timeUnit: A calendar component that tells the row grouping how
to create its groups.
    public init D          D              String
Calendar Component  where GroupingKey    Int  D
DataFrameProtocol


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension RowGrouping   RandomAccessCollection

    /// The index of the initial group in the row grouping.
    public var startIndex  Int   get

    /// The index of the final group in the row grouping.
    public var endIndex  Int   get
```

```
    /// Returns the index immediately after an element index.
    /// - Parameter i: A valid index to an element in the column.
    public func index           Int     Int

    /// Returns the index immediately before an element index.
    /// - Parameter i: A valid index to an element in the column.
    public func index            Int     Int

    /// The number of groups in the row grouping.
    public var count  Int   get

    /// Retrieves a group at an index.
    /// - Parameter position: A valid index to a group in the row grouping.
    public subscript           Int           GroupingKey
       DataFrame Slice    get

    /// A type representing the sequence's elements.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias Element         GroupingKey
DataFrame Slice

    /// A type that represents a position in the collection.
    ///
    /// Valid indices consist of the position of every element and a
    /// "past the end" position that's not valid for use as a subscript
    /// argument.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias Index   Int

    /// A type that represents the indices that are valid for subscripting the
    /// collection, in ascending order.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias Indices   Range Int

    /// A type that provides the collection's iteration interface and
    /// encapsulates its iteration state.
    ///
    /// By default, a collection conforms to the `Sequence` protocol by
    /// supplying `IndexingIterator` as its associated `Iterator`
    /// type.
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias Iterator
IndexingIterator RowGrouping GroupingKey

    /// A collection representing a contiguous subrange of this collection's
    /// elements. The subsequence shares indices with the original collection.
    ///
    /// The default subsequence type for collections that don't define their own
    /// is `Slice`.
```

```
    @available iOS 15  tvOS 15  watchOS 8  macOS 12
    public typealias SubSequence
Slice RowGrouping GroupingKey


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension RowGrouping

    /// Generates a group summaries instance for the columns of the row
grouping.
    public func summary      any GroupSummaries

    /// Generates a group summaries instance for the columns you select by
name.
    ///
    /// - Parameter columnNames: An array of column names.
    public func summary                  String      any
GroupSummaries


/// A type that represents a collection of row selections that have the same value
in a column.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public protocol RowGroupingProtocol   CustomStringConvertible


    /// The number of groups in the row grouping.
    var count  Int   get

    /// Generates a data frame that contains all the rows from each group in the
row grouping.
    ///
    /// A row grouping can only use this method if all its groups have the same
column names and types.
    ///
    /// > Important: The method discards a column with the same name as the
row grouping itself.
    func ungrouped      DataFrame

    /// Generates a data frame, that you choose how to sort, with two columns,
one that has a row for each group key and
    /// another for the number or rows in the group.
    ///
    /// - Parameter order: A sorting order the method uses to sort the data
frame by its count column.
    ///
    /// The name of the data frame's column that stores the number of rows in
each group is *count*.
    func counts        Order      DataFrame
```

```
    /// Generates a data frame by aggregating each group's contents for each column you select by name.
    ///
    /// - Parameters:
    ///   - columnNames: A comma-separated, or variadic, list of column names.
    ///   - naming: A closure that converts a column name to another name.
    ///   - transform: A closure that aggregates a group's elements in a specific column.
    ///
    /// The data frame contains two columns that:
    /// - Identify each group
    /// - Store the results of your aggregation transform closure
    func aggregated Element  Result                    String
         String     String
 DiscontiguousColumnSlice Element   throws    Result
rethrows    DataFrame

    /// Returns a row grouping containing only the groups that satisfy a predicate.
    ///
    /// - Parameter isIncluded: A predicate closure that takes a group and returns a Boolean that indicates whether
    ///   the group is included.
    /// - Returns: A data frame slice that contains the rows that satisfy the predicate.
    @available macOS 12.3  iOS 15.4  tvOS 15.4  watchOS 8.5

    func filter _              DataFrame Slice  throws
Bool  rethrows    Self

    /// Generates a row grouping that applies a transformation closure to each group in the original.
    ///
    /// - Parameter transform: A closure that generates a data frame from a data frame slice that represents a group.
    func mapGroups _             DataFrame Slice  throws
DataFrame  rethrows    Self

    /// Generates two row groupings by randomly splitting the original by a proportion.
    /// - Parameters:
    ///   - proportion: A proportion in the range `[0.0, 1.0]`.
    ///   - seed: A seed number for a random-number generator.
    /// - Returns: A tuple of two data row grouping types.
    func randomSplit                  Double        Int
 Self  Self

    /// Generates a group summaries instance of the row grouping's columns.
    func summary       any GroupSummaries
```

```
    /// Generates a group summaries instance of the row grouping's columns you select by name.
    ///
    /// - Parameter columnNames: An array of column names.
    func summary                    String      any GroupSummaries

    /// Retrieves a group slice by key.
    ///
    /// - Parameter keys: A comma-separated, or variadic, list of key optionals.
    subscript       Any         DataFrame Slice    get


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension RowGroupingProtocol

    /// Generates a data frame with two columns, one that has a row for each group key and another for the number of
    /// rows in the group.
    ///
    /// The name of the data frame's column that stores the number of rows in each group is *count*.
    public func counts      DataFrame

    /// Generates a data frame that contains the sum of each group's rows along a column you select
    /// by name.
    /// - Parameters:
    ///   - columnName: The name of a column.
    ///   - type: The type of the column.
    ///   - order: A sorting order the method uses to sort the data frame by its sum column.
    public func sums N  _             String  _       N    Order    nil     DataFrame where N AdditiveArithmetic  N   Comparable

    /// Generates a data frame that contains the sum of each group's rows along a column you select
    /// by column identifier.
    /// - Parameters:
    ///   - columnID: A column identifier.
    ///   - order: A sorting order the method uses to sort the data frame by its sum column.
    public func sums N  _           ColumnID N          Order  nil     DataFrame where N   AdditiveArithmetic  N   Comparable

    /// Generates a data frame that contains the average mean of each group's rows along a column you select
```

```
    /// by name.
    /// - Parameters:
    ///   - columnName: The name of a column.
    ///   - type: The type of the column.
    ///   - order: A sorting order the method uses to sort the data frame by its mean column.
    public func means N  _              String  _       N
       Order    nil     DataFrame where N   FloatingPoint

    /// Generates a data frame that contains the average mean of each group's rows along a column you select
    /// by column identifier.
    /// - Parameters:
    ///   - columnID: A column identifier.
    ///   - order: A sorting order the method uses to sort the data frame by its mean column.
    public func means N  _           ColumnID N
Order    nil     DataFrame where N   FloatingPoint

    /// Generates a data frame that contains the quantile of each group's rows along a column you select
    /// by name.
    /// - Parameters:
    ///   - columnName: The name of a column.
    ///   - type: The type of the column.
    ///   - quantile: A number between 0.0 and 1.0 that represents the percentage of the data that lies below
    ///     the resulting value.
    ///   - order: A sorting order the method uses to sort the data frame by its quantile column.
    /// - Precondition: Quantile must be between 0.0 and 1.0.
    public func quantiles N  _              String  _
N                 N         Order    nil     DataFrame where N
  BinaryFloatingPoint

    /// Generates a data frame that contains the quantiles of each group's rows along a column you select
    /// by column identifier.
    /// - Parameters:
    ///   - columnID: A column identifier.
    ///   - quantile: A number between 0.0 and 1.0 that represents the percentage of the data that lies below
    ///     the resulting value.
    ///   - order: A sorting order the method uses to sort the data frame by its third quartile column.
    /// - Precondition: Quantile must be between 0.0 and 1.0.
    public func quantiles N  _           ColumnID N
          N         Order    nil     DataFrame where N
BinaryFloatingPoint

    /// Generates a data frame that contains the minimums of each group's rows
```

```
along a column you select
    /// by name.
    /// - Parameters:
    ///   - columnName: The name of a column.
    ///   - type: The type of the column.
    ///   - order: A sorting order the method uses to sort the data frame by
its minimum column.
    public func minimums N  _             String  _
N             Order    nil    DataFrame where N  Comparable

    /// Generates a data frame that contains the minimums of each group's rows
along a column you select
    /// by column identifier.
    /// - Parameters:
    ///   - columnID: A column identifier.
    ///   - order: A sorting order the method uses to sort the data frame by
its minimum column.
    public func minimums N  _           ColumnID N
Order    nil    DataFrame where N  Comparable

    /// Generates a data frame that contains the maximums of each group's rows
along a column you select
    /// by name.
    /// - Parameters:
    ///   - columnName: The name of a column.
    ///   - type: The type of the column.
    ///   - order: A sorting order the method uses to sort the data frame by
its maximum column.
    public func maximums N  _             String  _
N             Order    nil    DataFrame where N  Comparable

    /// Generates a data frame that contains the maximums of each group's rows
along a column you select
    /// by column identifier.
    /// - Parameters:
    ///   - columnID: A column identifier.
    ///   - order: A sorting order the method uses to sort the data frame by
its maximum column.
    public func maximums N  _           ColumnID N
Order    nil    DataFrame where N  Comparable

    /// Generates a data frame by aggregating each group's contents for each
column you select by name.
    ///
    /// - Parameters:
    ///   - columnNames: A comma-separated, or variadic, list of column
names.
    ///   - naming: A closure that converts a column name to another name.
    ///   - transform: A closure that aggregates a group's elements in a
specific column.
    ///
```

```
    /// The data frame contains two columns that:
    /// - Identify each group
    /// - Store the results of your aggregation transform closure
    public func aggregated Element  Result
String              String     String
 DiscontiguousColumnSlice Element   throws    Result
rethrows    DataFrame

    /// Generates a data frame with a column for the group identifier and a
column of values from the transform.
    ///
    ///
    /// - Parameters:
    ///   - columnID: A column identifier.
    ///   - aggregatedColumnName: The name of the aggregation column
the method adds to the data frame.
    ///   - transform: A closure that transforms each group's elements in
the column.
    public func aggregated Element  Result
ColumnID Element                              String    nil
           DiscontiguousColumnSlice Element   throws
Result  rethrows    DataFrame

    /// Generates two row groupings by randomly splitting the original with a
proportion.
    /// - Parameters proportion: A proportion in the range `[0.0, 1.0]`.
    /// - Returns: A tuple of two row groupings.
    public func randomSplit                  Double      Self
Self


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension RowGroupingProtocol

    /// Generates a categorical summary of the columns you select by name.
    ///
    /// - Parameter columnNames: A comma-separated, or variadic, list of
column names.
    ///
    /// - Returns: A ``GroupSummaries`` instance that contains
categorical summary of the columns you select.
    public func summary                  String        any
GroupSummaries


/// An error when reading a Turi Create scalable data frame.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public enum SFrameReadingError   Error

    /// An error that indicates the scalable data frame directory is missing an
```

```
archive file.
    case missingArchive

    /// An error that indicates the scalable data frame directory's archive file is
corrupt.
    ///
    /// The associated value contains a description of the error.
    case badArchive String

    /// An error that indicates the scalable data frame contains an archive
version or layout the framework doesn't
    /// support.
    ///
    /// The associated value contains a description of the error.
    case unsupportedArchive String

    /// An error that indicates the scalable data frame contains an unknown or
unsupported data type.
    ///
    /// The associated value contains the unknown data type identifier.
    case unsupportedType Int

    /// An error that indicates the scalable data frame contains an unsupported
data layout.
    ///
    /// The associated value contains a description of the error.
    case unsupportedLayout String

    /// An error that indicates the scalable data frame contains bad data
encoding.
    ///
    /// The associated value contains a description of the error.
    case badEncoding String

    /// An error that indicates the scalable data frame is missing one of the
requested columns.
    ///
    /// The associated value contains a description of the error.
    case missingColumn String


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension SFrameReadingError  CustomStringConvertible

    /// A text representation of the error.
    public var description  String  get


@available macOS 14  iOS 17  tvOS 17  watchOS 10
extension SFrameReadingError  LocalizedError
```

```
    /// A localized message describing what error occurred.
    public var errorDescription  String    get


/// A collection type that represents multidimensional data in a data frame element.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public struct ShapedData Element

    /// An integer array that stores the size of each dimension in the
corresponding element.
    public let shape   Int

    /// An integer array that stores the number of memory locations
    /// that span the length of each dimension in the corresponding element.
    public let strides   Int

    /// A linear array that stores the elements of the multidimensional array.
    public let contents   Element

    /// Creates a multidimensional shaped array from a one-dimensional array.
    /// - Parameters:
    ///   - shape: An integer array that stores the size of each dimension in
the corresponding element.
    ///   - strides: An integer array that stores the number of memory
locations
    ///   that span the length of each dimension in the corresponding element.
    ///   - contents: A linear array that stores the elements of the
multidimensional array.
    public init         Int             Int
 Element

    /// Retrieves an element using an index for each dimension.
    /// - Parameter indices: A comma-separated, or variadic, list of
indices, with one for each dimension.
    public subscript          Int        Element   get


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension ShapedData   Equatable where Element   Equatable

    /// Returns a Boolean value indicating whether two values are equal.
    ///
    /// Equality is the inverse of inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is `false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to compare.
    public static func        ShapedData Element
```

```
ShapedData Element      Bool


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension ShapedData   Hashable where Element   Hashable

    /// Hashes the essential components of this value by feeding them into the
    /// given hasher.
    ///
    /// Implement this method to conform to the `Hashable` protocol. The
    /// components used for hashing must be the same as the components
compared
    /// in your type's `==` operator implementation. Call
`hasher.combine(_:)`
    /// with each of these components.
    ///
    /// - Important: In your implementation of `hash(into:)`,
    ///    don't call `finalize()` on the `hasher` instance provided,
    ///    or replace it with a different instance.
    ///    Doing so may become a compile-time error in the future.
    ///
    /// - Parameter hasher: The hasher to use when combining the
components
    ///    of this instance.
    public func hash               inout Hasher

    /// The hash value.
    ///
    /// Hash values are not guaranteed to be equal across different executions of
    /// your program. Do not save hash values to use during a future execution.
    ///
    /// - Important: `hashValue` is deprecated as a `Hashable`
requirement. To
    ///    conform to `Hashable`, implement the `hash(into:)` requirement
instead.
    ///    The compiler provides an implementation for `hashValue` for you.
    public var hashValue  Int   get


@available macOS 12  iOS 15  tvOS 15  watchOS 8
extension ShapedData   Sendable where Element   Sendable


/// The summary data frame column identifiers.
@available macOS 12  iOS 15  tvOS 15  watchOS 8
public enum SummaryColumnIDs   Sendable

    /// The identifier that represents the summary's column that contains the
column names in a data frame.
    public static let columnName  ColumnID String
```

```
    /// The identifier that represents the summary's column of arithmetic means.
    public static let mean = ColumnID<Double>()

    /// The identifier that represents the summary's column of standard deviations.
    public static let standardDeviation = ColumnID<Double>()

    /// The identifier that represents the summary's column of minimums.
    public static let minimum = ColumnID<Double>()

    /// The identifier that represents the summary's column of maximums.
    public static let maximum = ColumnID<Double>()

    /// The identifier that represents the summary's column of medians.
    public static let median = ColumnID<Double>()

    /// The identifier that represents the summary's column of first quartiles.
    public static let firstQuartile = ColumnID<Double>()

    /// The identifier that represents the summary's column of third quartiles.
    public static let thirdQuartile = ColumnID<Double>()

    /// The identifier that represents the summary's column of most frequent elements.
    public static let mode = ColumnID<Any>()

    /// The identifier that represents the summary's column of unique counts.
    public static let uniqueCount = ColumnID<Int>()

    /// The identifier that represents the summary's column of missing counts.
    public static let noneCount = ColumnID<Int>()

    /// The identifier that represents the summary's column of non-missing counts.
    public static let someCount = ColumnID<Int>()
```