Other API changes

- A newly constructed empty *DataFrame* with integer as the `dtype` will now only be cast to `float64` if `index` is specified (GH22858)
- *Series.str.cat()* will now raise if `others` is a `set` (GH23009)
- Passing scalar values to *DatetimeIndex* or *TimedeltaIndex* will now raise `TypeError` instead of `ValueError` (GH23539)
- `max_rows` and `max_cols` parameters removed from `HTMLFormatter` since truncation is handled by `DataFrameFormatter` (GH23818)
- *read_csv()* will now raise a `ValueError` if a column with missing values is declared as having dtype `bool` (GH20591)
- The column order of the resultant *DataFrame* from *MultiIndex.to_frame()* is now guaranteed to match the *MultiIndex.names* order. (GH22420)
- Incorrectly passing a *DatetimeIndex* to *MultiIndex.from_tuples()*, rather than a sequence of tuples, now raises a `TypeError` rather than a `ValueError` (GH24024)
- `pd.offsets.generate_range()` argument `time_rule` has been removed; use `offset` instead (GH24157)
- In 0.23.x, pandas would raise a `ValueError` on a merge of a numeric column (e.g. `int` dtyped column) and an `object` dtyped column (GH9780). We have re-enabled the ability to merge `object` and other dtypes; pandas will still raise on a merge between a numeric and an `object` dtyped column that is composed only of strings (GH21681)
- Accessing a level of a `MultiIndex` with a duplicate name (e.g. in *get_level_values()*) now raises a `ValueError` instead of a `KeyError` (GH21678).
- Invalid construction of `IntervalDtype` will now always raise a `TypeError` rather than a `ValueError` if the subdtype is invalid (GH21185)
- Trying to reindex a `DataFrame` with a non unique `MultiIndex` now raises a `ValueError` instead of an `Exception` (GH21770)
- *Index* subtraction will attempt to operate element-wise instead of raising `TypeError` (GH19369)
- *pandas.io.formats.style.Styler* supports a `number-format` property when using *to_excel()* (GH22015)
- *DataFrame.corr()* and *Series.corr()* now raise a `ValueError` along with a helpful error message instead of a `KeyError` when supplied with an invalid method (GH22298)
- `shift()` will now always return a copy, instead of the previous behaviour of returning self when shifting by 0 (GH22397)
- *DataFrame.set_index()* now gives a better (and less frequent) KeyError, raises a `ValueError` for incorrect types, and will not fail on duplicate column names with `drop=True`. (GH22484)
- Slicing a single row of a DataFrame with multiple ExtensionArrays of the same type now preserves the dtype, rather than coercing to object (GH22784)
- `DateOffset` attribute *_cacheable* and method *_should_cache* have been removed (GH23118)
- *Series.searchsorted()*, when supplied a scalar value to search for, now returns a scalar instead of an array (GH23801).
- `Categorical.searchsorted()`, when supplied a scalar value to search for, now returns a scalar instead of an array (GH23466).

- `Categorical.searchsorted()` now raises a `KeyError` rather that a `ValueError`, if a searched for key is not found in its categories (GH23466).
- *`Index.hasnans()`* and *`Series.hasnans()`* now always return a python boolean. Previously, a python or a numpy boolean could be returned, depending on circumstances (GH23294).
- The order of the arguments of *`DataFrame.to_html()`* and *`DataFrame.to_string()`* is rearranged to be consistent with each other. (GH23614)
- `CategoricalIndex.reindex()` now raises a `ValueError` if the target index is non-unique and not equal to the current index. It previously only raised if the target index was not of a categorical dtype (GH23963).
- *`Series.to_list()`* and *`Index.to_list()`* are now aliases of `Series.tolist` respectively `Index.tolist` (GH8826)
- The result of `SparseSeries.unstack` is now a *`DataFrame`* with sparse values, rather than a `SparseDataFrame` (GH24372).
- *`DatetimeIndex`* and *`TimedeltaIndex`* no longer ignore the dtype precision. Passing a non-nanosecond resolution dtype will raise a `ValueError` (GH24753)

Extension type changes

Equality and hashability

Pandas now requires that extension dtypes be hashable (i.e. the respective `ExtensionDtype` objects; hashability is not a requirement for the values of the corresponding `ExtensionArray`). The base class implements a default `__eq__` and `__hash__`. If you have a parametrized dtype, you should update the `ExtensionDtype._metadata` tuple to match the signature of your `__init__` method. See *`pandas.api.extensions.ExtensionDtype`* for more (GH22476).

New and changed methods

- `dropna()` has been added (GH21185)
- `repeat()` has been added (GH24349)
- The `ExtensionArray` constructor, `_from_sequence` now take the keyword arg `copy=False` (GH21185)
- *`pandas.api.extensions.ExtensionArray.shift()`* added as part of the basic `ExtensionArray` interface (GH22387).
- `searchsorted()` has been added (GH24350)
- Support for reduction operations such as `sum`, `mean` via opt-in base class method override (GH22762)
- `ExtensionArray.isna()` is allowed to return an `ExtensionArray` (GH22325).

Dtype changes

- `ExtensionDtype` has gained the ability to instantiate from string dtypes, e.g. `decimal` would instantiate a registered `DecimalDtype`; furthermore the `ExtensionDtype` has gained the method `construct_array_type` (GH21185)
- Added `ExtensionDtype._is_numeric` for controlling whether an extension dtype is considered numeric (GH22290).
- Added `pandas.api.types.register_extension_dtype()` to register an extension type with pandas (GH22664)
- Updated the `.type` attribute for `PeriodDtype`, `DatetimeTZDtype`, and `IntervalDtype` to be instances of the dtype (`Period`, `Timestamp`, and `Interval` respectively) (GH22938)

Operator support

A `Series` based on an `ExtensionArray` now supports arithmetic and comparison operators (GH19577). There are two approaches for providing operator support for an `ExtensionArray`:

1. Define each of the operators on your `ExtensionArray` subclass.
2. Use an operator implementation from pandas that depends on operators that are already defined on the underlying elements (scalars) of the `ExtensionArray`.

See the *ExtensionArray Operator Support* documentation section for details on both ways of adding operator support.

Other changes

- A default repr for `pandas.api.extensions.ExtensionArray` is now provided (GH23601).
- `ExtensionArray._formatting_values()` is deprecated. Use `ExtensionArray._formatter` instead. (GH23601)
- An `ExtensionArray` with a boolean dtype now works correctly as a boolean indexer. `pandas.api.types.is_bool_dtype()` now properly considers them boolean (GH22326)

Bug fixes

- Bug in `Series.get()` for `Series` using `ExtensionArray` and integer index (GH21257)
- `shift()` now dispatches to `ExtensionArray.shift()` (GH22386)
- `Series.combine()` works correctly with `ExtensionArray` inside of `Series` (GH20825)
- `Series.combine()` with scalar argument now works for any function type (GH21248)
- `Series.astype()` and `DataFrame.astype()` now dispatch to `ExtensionArray.astype()` (GH21185).
- Slicing a single row of a `DataFrame` with multiple ExtensionArrays of the same type now preserves the dtype, rather than coercing to object (GH22784)
- Bug when concatenating multiple `Series` with different extension dtypes not casting to object dtype (GH22994)
- Series backed by an `ExtensionArray` now work with `util.hash_pandas_object()` (GH23066)
- `DataFrame.stack()` no longer converts to object dtype for DataFrames where each column has the same extension dtype. The output Series will have the same dtype as the columns (GH23077).
- `Series.unstack()` and `DataFrame.unstack()` no longer convert extension arrays to object-dtype ndarrays. Each column in the output `DataFrame` will now have the same dtype as the input (GH23077).
- Bug when grouping `Dataframe.groupby()` and aggregating on `ExtensionArray` it was not returning the actual `ExtensionArray` dtype (GH23227).
- Bug in `pandas.merge()` when merging on an extension array-backed column (GH23020).

Deprecations

- `MultiIndex.labels` has been deprecated and replaced by `MultiIndex.codes`. The functionality is unchanged. The new name better reflects the natures of these codes and makes the `MultiIndex` API more similar to the API for `CategoricalIndex` (GH13443). As a consequence, other uses of the name `labels` in `MultiIndex` have also been deprecated and replaced with `codes`:
  - You should initialize a `MultiIndex` instance using a parameter named `codes` rather than `labels`.
  - `MultiIndex.set_labels` has been deprecated in favor of `MultiIndex.set_codes()`.
  - For method `MultiIndex.copy()`, the `labels` parameter has been deprecated and replaced by a `codes` parameter.
- `DataFrame.to_stata()`, `read_stata()`, `StataReader` and `StataWriter` have deprecated the `encoding` argument. The encoding of a Stata dta file is determined by the file type and cannot be changed (GH21244)
- `MultiIndex.to_hierarchical()` is deprecated and will be removed in a future version (GH21613)
- `Series.ptp()` is deprecated. Use `numpy.ptp` instead (GH21614)
- `Series.compress()` is deprecated. Use `Series[condition]` instead (GH18262)
- The signature of `Series.to_csv()` has been uniformed to that of `DataFrame.to_csv()`: the name of the first argument is now `path_or_buf`, the order of subsequent arguments has changed, the `header` argument now defaults to `True`. (GH19715)
- `Categorical.from_codes()` has deprecated providing float values for the `codes` argument. (GH21767)
- `pandas.read_table()` is deprecated. Instead, use `read_csv()` passing `sep='\t'` if necessary. This deprecation has been removed in 0.25.0. (GH21948)
- `Series.str.cat()` has deprecated using arbitrary list-likes *within* list-likes. A list-like container may still contain many `Series`, `Index` or 1-dimensional `np.ndarray`, or alternatively, only scalar values. (GH21950)
- `FrozenNDArray.searchsorted()` has deprecated the `v` parameter in favor of `value` (GH14645)
- `DatetimeIndex.shift()` and `PeriodIndex.shift()` now accept `periods` argument instead of `n` for consistency with `Index.shift()` and `Series.shift()`. Using `n` throws a deprecation warning (GH22458, GH22912)
- The `fastpath` keyword of the different Index constructors is deprecated (GH23110).
- `Timestamp.tz_localize()`, `DatetimeIndex.tz_localize()`, and `Series.tz_localize()` have deprecated the `errors` argument in favor of the `nonexistent` argument (GH8917)
- The class `FrozenNDArray` has been deprecated. When unpickling, `FrozenNDArray` will be unpickled to `np.ndarray` once this class is removed (GH9031)
- The methods `DataFrame.update()` and `Panel.update()` have deprecated the `raise_conflict=False|True` keyword in favor of `errors='ignore'|'raise'` (GH23585)
- The methods `Series.str.partition()` and `Series.str.rpartition()` have deprecated the `pat` keyword in favor of `sep` (GH22676)
- Deprecated the `nthreads` keyword of `pandas.read_feather()` in favor of `use_threads` to reflect the changes in `pyarrow>=0.11.0`. (GH23053)
- `pandas.read_excel()` has deprecated accepting `usecols` as an integer. Please pass in a list of ints from 0 to `usecols` inclusive instead (GH23527)

- Constructing a *TimedeltaIndex* from data with datetime64-dtyped data is deprecated, will raise TypeError in a future version (GH23539)
- Constructing a *DatetimeIndex* from data with timedelta64-dtyped data is deprecated, will raise TypeError in a future version (GH23675)
- The keep_tz=False option (the default) of the keep_tz keyword of *DatetimeIndex.to_series()* is deprecated (GH17832).
- Timezone converting a tz-aware datetime.datetime or *Timestamp* with *Timestamp* and the tz argument is now deprecated. Instead, use *Timestamp.tz_convert()* (GH23579)
- pandas.api.types.is_period() is deprecated in favor of pandas.api.types.is_period_dtype (GH23917)
- pandas.api.types.is_datetimetz() is deprecated in favor of pandas.api.types.is_datetime64tz (GH23917)
- Creating a *TimedeltaIndex*, *DatetimeIndex*, or *PeriodIndex* by passing range arguments *start*, *end*, and *periods* is deprecated in favor of *timedelta_range()*, *date_range()*, or *period_range()* (GH23919)
- Passing a string alias like 'datetime64[ns, UTC]' as the unit parameter to *DatetimeTZDtype* is deprecated. Use DatetimeTZDtype.construct_from_string instead (GH23990).
- The skipna parameter of *infer_dtype()* will switch to True by default in a future version of pandas (GH17066, GH24050)
- In *Series.where()* with Categorical data, providing an other that is not present in the categories is deprecated. Convert the categorical to a different dtype or add the other to the categories first (GH24077).
- Series.clip_lower(), Series.clip_upper(), DataFrame.clip_lower() and DataFrame.clip_upper() are deprecated and will be removed in a future version. Use Series.clip(lower=threshold), Series.clip(upper=threshold) and the equivalent DataFrame methods (GH24203)
- Series.nonzero() is deprecated and will be removed in a future version (GH18262)
- Passing an integer to *Series.fillna()* and *DataFrame.fillna()* with timedelta64[ns] dtypes is deprecated, will raise TypeError in a future version. Use obj.fillna(pd.Timedelta(...)) instead (GH24694)
- Series.cat.categorical, Series.cat.name and Series.cat.index have been deprecated. Use the attributes on Series.cat or Series directly. (GH24751).
- Passing a dtype without a precision like np.dtype('datetime64') or timedelta64 to *Index*, *DatetimeIndex* and *TimedeltaIndex* is now deprecated. Use the nanosecond-precision dtype instead (GH24753).

Integer addition/subtraction with datetimes and timedeltas is deprecated

In the past, users could—in some cases—add or subtract integers or integer-dtype arrays from *Timestamp*, *DatetimeIndex* and *TimedeltaIndex*.

This usage is now deprecated. Instead add or subtract integer multiples of the object's freq attribute (GH21939, GH23878).

Previous behavior:

```
In [5]: ts = pd.Timestamp('1994-05-06 12:15:16', freq=pd.offsets.Hour())
In [6]: ts + 2
Out[6]: Timestamp('1994-05-06 14:15:16', freq='H')

In [7]: tdi = pd.timedelta_range('1D', periods=2)
In [8]: tdi - np.array([2, 1])
Out[8]: TimedeltaIndex(['-1 days', '1 days'], dtype='timedelta64[ns]', freq=None)

In [9]: dti = pd.date_range('2001-01-01', periods=2, freq='7D')
In [10]: dti + pd.Index([1, 2])
Out[10]: DatetimeIndex(['2001-01-08', '2001-01-22'], dtype='datetime64[ns]',␣
↪freq=None)
```

New behavior:

```
In [108]: ts = pd.Timestamp('1994-05-06 12:15:16', freq=pd.offsets.Hour())

In [109]: ts + 2 * ts.freq
Out[109]: Timestamp('1994-05-06 14:15:16', freq='H')

In [110]: tdi = pd.timedelta_range('1D', periods=2)

In [111]: tdi - np.array([2 * tdi.freq, 1 * tdi.freq])
Out[111]: TimedeltaIndex(['-1 days', '1 days'], dtype='timedelta64[ns]', freq=None)

In [112]: dti = pd.date_range('2001-01-01', periods=2, freq='7D')

In [113]: dti + pd.Index([1 * dti.freq, 2 * dti.freq])
Out[113]: DatetimeIndex(['2001-01-08', '2001-01-22'], dtype='datetime64[ns]',␣
↪freq=None)
```

Passing integer data and a timezone to datetimeindex

The behavior of `DatetimeIndex` when passed integer data and a timezone is changing in a future version of pandas. Previously, these were interpreted as wall times in the desired timezone. In the future, these will be interpreted as wall times in UTC, which are then converted to the desired timezone (GH24559).

The default behavior remains the same, but issues a warning:

```
In [3]: pd.DatetimeIndex([946684800000000000], tz="US/Central")
/bin/ipython:1: FutureWarning:
    Passing integer-dtype data and a timezone to DatetimeIndex. Integer values
    will be interpreted differently in a future version of pandas. Previously,
    these were viewed as datetime64[ns] values representing the wall time
    *in the specified timezone*. In the future, these will be viewed as
    datetime64[ns] values representing the wall time *in UTC*. This is similar
    to a nanosecond-precision UNIX epoch. To accept the future behavior, use

        pd.to_datetime(integer_data, utc=True).tz_convert(tz)

    To keep the previous behavior, use

        pd.to_datetime(integer_data).tz_localize(tz)

 #!/bin/python3
 Out[3]: DatetimeIndex(['2000-01-01 00:00:00-06:00'], dtype='datetime64[ns, US/
↪Central]', freq=None)
```

(continues on next page)

(continued from previous page)

As the warning message explains, opt in to the future behavior by specifying that the integer values are UTC, and then converting to the final timezone:

```
In [114]: pd.to_datetime([946684800000000000], utc=True).tz_convert('US/Central')
Out[114]: DatetimeIndex(['1999-12-31 18:00:00-06:00'], dtype='datetime64[ns, US/
↪Central]', freq=None)
```

The old behavior can be retained with by localizing directly to the final timezone:

```
In [115]: pd.to_datetime([946684800000000000]).tz_localize('US/Central')
Out[115]: DatetimeIndex(['2000-01-01 00:00:00-06:00'], dtype='datetime64[ns, US/
↪Central]', freq=None)
```

Converting timezone-aware Series and Index to NumPy arrays

The conversion from a `Series` or `Index` with timezone-aware datetime data will change to preserve timezones by default (GH23569).

NumPy doesn't have a dedicated dtype for timezone-aware datetimes. In the past, converting a `Series` or `DatetimeIndex` with timezone-aware datatimes would convert to a NumPy array by

1. converting the tz-aware data to UTC
2. dropping the timezone-info
3. returning a `numpy.ndarray` with `datetime64[ns]` dtype

Future versions of pandas will preserve the timezone information by returning an object-dtype NumPy array where each value is a `Timestamp` with the correct timezone attached

```
In [116]: ser = pd.Series(pd.date_range('2000', periods=2, tz="CET"))

In [117]: ser
Out[117]:
0   2000-01-01 00:00:00+01:00
1   2000-01-02 00:00:00+01:00
Length: 2, dtype: datetime64[ns, CET]
```

The default behavior remains the same, but issues a warning

```
In [8]: np.asarray(ser)
/bin/ipython:1: FutureWarning: Converting timezone-aware DatetimeArray to timezone-
↪naive
      ndarray with 'datetime64[ns]' dtype. In the future, this will return an ndarray
      with 'object' dtype where each element is a 'pandas.Timestamp' with the correct
↪'tz'.

        To accept the future behavior, pass 'dtype=object'.
        To keep the old behavior, pass 'dtype="datetime64[ns]"'.
  #!/bin/python3
Out[8]:
array(['1999-12-31T23:00:00.000000000', '2000-01-01T23:00:00.000000000'],
      dtype='datetime64[ns]')
```

The previous or future behavior can be obtained, without any warnings, by specifying the `dtype`

Previous behavior

```
In [118]: np.asarray(ser, dtype='datetime64[ns]')
Out[118]:
array(['1999-12-31T23:00:00.000000000', '2000-01-01T23:00:00.000000000'],
      dtype='datetime64[ns]')
```

Future behavior

```
# New behavior
In [119]: np.asarray(ser, dtype=object)
Out[119]:
array([Timestamp('2000-01-01 00:00:00+0100', tz='CET', freq='D'),
       Timestamp('2000-01-02 00:00:00+0100', tz='CET', freq='D')],
      dtype=object)
```

Or by using *Series.to_numpy()*

```
In [120]: ser.to_numpy()
Out[120]:
array([Timestamp('2000-01-01 00:00:00+0100', tz='CET', freq='D'),
       Timestamp('2000-01-02 00:00:00+0100', tz='CET', freq='D')],
      dtype=object)

In [121]: ser.to_numpy(dtype="datetime64[ns]")
Out[121]:
array(['1999-12-31T23:00:00.000000000', '2000-01-01T23:00:00.000000000'],
      dtype='datetime64[ns]')
```

All the above applies to a *DatetimeIndex* with tz-aware values as well.

Removal of prior version deprecations/changes

- The `LongPanel` and `WidePanel` classes have been removed (GH10892)
- *Series.repeat()* has renamed the `reps` argument to `repeats` (GH14645)
- Several private functions were removed from the (non-public) module `pandas.core.common` (GH22001)
- Removal of the previously deprecated module `pandas.core.datetools` (GH14105, GH14094)
- Strings passed into *DataFrame.groupby()* that refer to both column and index levels will raise a `ValueError` (GH14432)
- *Index.repeat()* and `MultiIndex.repeat()` have renamed the `n` argument to `repeats` (GH14645)
- The `Series` constructor and `.astype` method will now raise a `ValueError` if timestamp dtypes are passed in without a unit (e.g. `np.datetime64`) for the `dtype` parameter (GH15987)
- Removal of the previously deprecated `as_indexer` keyword completely from `str.match()` (GH22356, GH6581)
- The modules `pandas.types`, `pandas.computation`, and `pandas.util.decorators` have been removed (GH16157, GH16250)
- Removed the `pandas.formats.style` shim for *pandas.io.formats.style.Styler* (GH16059)
- `pandas.pnow`, `pandas.match`, `pandas.groupby`, `pd.get_store`, `pd.Expr`, and `pd.Term` have been removed (GH15538, GH15940)

- `Categorical.searchsorted()` and *`Series.searchsorted()`* have renamed the `v` argument to `value` (GH14645)
- `pandas.parser`, `pandas.lib`, and `pandas.tslib` have been removed (GH15537)
- *`Index.searchsorted()`* have renamed the `key` argument to `value` (GH14645)
- `DataFrame.consolidate` and `Series.consolidate` have been removed (GH15501)
- Removal of the previously deprecated module `pandas.json` (GH19944)
- The module `pandas.tools` has been removed (GH15358, GH16005)
- `SparseArray.get_values()` and `SparseArray.to_dense()` have dropped the `fill` parameter (GH14686)
- `DataFrame.sortlevel` and `Series.sortlevel` have been removed (GH15099)
- `SparseSeries.to_dense()` has dropped the `sparse_only` parameter (GH14686)
- *`DataFrame.astype()`* and *`Series.astype()`* have renamed the `raise_on_error` argument to `errors` (GH14967)
- `is_sequence`, `is_any_int_dtype`, and `is_floating_dtype` have been removed from `pandas.api.types` (GH16163, GH16189)

Performance improvements

- Slicing Series and DataFrames with an monotonically increasing *`CategoricalIndex`* is now very fast and has speed comparable to slicing with an `Int64Index`. The speed increase is both when indexing by label (using .loc) and position(.iloc) (GH20395) Slicing a monotonically increasing *`CategoricalIndex`* itself (i.e. `ci[1000:2000]`) shows similar speed improvements as above (GH21659)
- Improved performance of *`CategoricalIndex.equals()`* when comparing to another *`CategoricalIndex`* (GH24023)
- Improved performance of *`Series.describe()`* in case of numeric dtpyes (GH21274)
- Improved performance of *`pandas.core.groupby.GroupBy.rank()`* when dealing with tied rankings (GH21237)
- Improved performance of *`DataFrame.set_index()`* with columns consisting of *`Period`* objects (GH21582, GH21606)
- Improved performance of *`Series.at()`* and *`Index.get_value()`* for Extension Arrays values (e.g. *`Categorical`*) (GH24204)
- Improved performance of membership checks in *`Categorical`* and *`CategoricalIndex`* (i.e. `x in cat`-style checks are much faster). `CategoricalIndex.contains()` is likewise much faster (GH21369, GH21508)
- Improved performance of *`HDFStore.groups()`* (and dependent functions like *`HDFStore.keys()`*. (i.e. `x in store` checks are much faster) (GH21372)
- Improved the performance of *`pandas.get_dummies()`* with `sparse=True` (GH21997)
- Improved performance of `IndexEngine.get_indexer_non_unique()` for sorted, non-unique indexes (GH9466)
- Improved performance of `PeriodIndex.unique()` (GH23083)
- Improved performance of *`concat()`* for *Series* objects (GH23404)

- Improved performance of `DatetimeIndex.normalize()` and `Timestamp.normalize()` for timezone naive or UTC datetimes (GH23634)
- Improved performance of `DatetimeIndex.tz_localize()` and various `DatetimeIndex` attributes with dateutil UTC timezone (GH23772)
- Fixed a performance regression on Windows with Python 3.7 of `read_csv()` (GH23516)
- Improved performance of `Categorical` constructor for `Series` objects (GH23814)
- Improved performance of `where()` for Categorical data (GH24077)
- Improved performance of iterating over a `Series`. Using `DataFrame.itertuples()` now creates iterators without internally allocating lists of all elements (GH20783)
- Improved performance of `Period` constructor, additionally benefitting `PeriodArray` and `PeriodIndex` creation (GH24084, GH24118)
- Improved performance of tz-aware `DatetimeArray` binary operations (GH24491)

Bug fixes

Categorical

- Bug in `Categorical.from_codes()` where `NaN` values in `codes` were silently converted to `0` (GH21767). In the future this will raise a `ValueError`. Also changes the behavior of `.from_codes([1.1, 2.0])`.
- Bug in `Categorical.sort_values()` where `NaN` values were always positioned in front regardless of `na_position` value. (GH22556).
- Bug when indexing with a boolean-valued `Categorical`. Now a boolean-valued `Categorical` is treated as a boolean mask (GH22665)
- Constructing a `CategoricalIndex` with empty values and boolean categories was raising a `ValueError` after a change to dtype coercion (GH22702).
- Bug in `Categorical.take()` with a user-provided `fill_value` not encoding the `fill_value`, which could result in a `ValueError`, incorrect results, or a segmentation fault (GH23296).
- In `Series.unstack()`, specifying a `fill_value` not present in the categories now raises a `TypeError` rather than ignoring the `fill_value` (GH23284)
- Bug when resampling `DataFrame.resample()` and aggregating on categorical data, the categorical dtype was getting lost. (GH23227)
- Bug in many methods of the `.str`-accessor, which always failed on calling the `CategoricalIndex.str` constructor (GH23555, GH23556)
- Bug in `Series.where()` losing the categorical dtype for categorical data (GH24077)
- Bug in `Categorical.apply()` where `NaN` values could be handled unpredictably. They now remain unchanged (GH24241)
- Bug in `Categorical` comparison methods incorrectly raising `ValueError` when operating against a `DataFrame` (GH24630)
- Bug in `Categorical.set_categories()` where setting fewer new categories with `rename=True` caused a segmentation fault (GH24675)

Datetimelike

- Fixed bug where two `DateOffset` objects with different `normalize` attributes could evaluate as equal (GH21404)
- Fixed bug where *`Timestamp.resolution()`* incorrectly returned 1-microsecond `timedelta` instead of 1-nanosecond *`Timedelta`* (GH21336, GH21365)
- Bug in *`to_datetime()`* that did not consistently return an *`Index`* when `box=True` was specified (GH21864)
- Bug in *`DatetimeIndex`* comparisons where string comparisons incorrectly raises `TypeError` (GH22074)
- Bug in *`DatetimeIndex`* comparisons when comparing against `timedelta64[ns]` dtyped arrays; in some cases `TypeError` was incorrectly raised, in others it incorrectly failed to raise (GH22074)
- Bug in *`DatetimeIndex`* comparisons when comparing against object-dtyped arrays (GH22074)
- Bug in *`DataFrame`* with `datetime64[ns]` dtype addition and subtraction with `Timedelta`-like objects (GH22005, GH22163)
- Bug in *`DataFrame`* with `datetime64[ns]` dtype addition and subtraction with `DateOffset` objects returning an `object` dtype instead of `datetime64[ns]` dtype (GH21610, GH22163)
- Bug in *`DataFrame`* with `datetime64[ns]` dtype comparing against `NaT` incorrectly (GH22242, GH22163)
- Bug in *`DataFrame`* with `datetime64[ns]` dtype subtracting `Timestamp`-like object incorrectly returned `datetime64[ns]` dtype instead of `timedelta64[ns]` dtype (GH8554, GH22163)
- Bug in *`DataFrame`* with `datetime64[ns]` dtype subtracting `np.datetime64` object with non-nanosecond unit failing to convert to nanoseconds (GH18874, GH22163)
- Bug in *`DataFrame`* comparisons against `Timestamp`-like objects failing to raise `TypeError` for inequality checks with mismatched types (GH8932, GH22163)
- Bug in *`DataFrame`* with mixed dtypes including `datetime64[ns]` incorrectly raising `TypeError` on equality comparisons (GH13128, GH22163)
- Bug in *`DataFrame.values`* returning a *`DatetimeIndex`* for a single-column `DataFrame` with tz-aware datetime values. Now a 2-D `numpy.ndarray` of *`Timestamp`* objects is returned (GH24024)
- Bug in *`DataFrame.eq()`* comparison against `NaT` incorrectly returning `True` or `NaN` (GH15697, GH22163)
- Bug in *`DatetimeIndex`* subtraction that incorrectly failed to raise `OverflowError` (GH22492, GH22508)
- Bug in *`DatetimeIndex`* incorrectly allowing indexing with `Timedelta` object (GH20464)
- Bug in *`DatetimeIndex`* where frequency was being set if original frequency was `None` (GH22150)
- Bug in rounding methods of *`DatetimeIndex`* (*`round()`*, *`ceil()`*, *`floor()`*) and *`Timestamp`* (*`round()`*, *`ceil()`*, *`floor()`*) could give rise to loss of precision (GH22591)
- Bug in *`to_datetime()`* with an *`Index`* argument that would drop the `name` from the result (GH21697)
- Bug in *`PeriodIndex`* where adding or subtracting a `timedelta` or `Tick` object produced incorrect results (GH22988)
- Bug in the *`Series`* repr with period-dtype data missing a space before the data (GH23601)
- Bug in *`date_range()`* when decrementing a start date to a past end date by a negative frequency (GH23270)
- Bug in *`Series.min()`* which would return `NaN` instead of `NaT` when called on a series of `NaT` (GH23282)
- Bug in *`Series.combine_first()`* not properly aligning categoricals, so that missing values in `self` where not filled by valid values from `other` (GH24147)

- Bug in `DataFrame.combine()` with datetimelike values raising a TypeError (GH23079)
- Bug in `date_range()` with frequency of `Day` or higher where dates sufficiently far in the future could wrap around to the past instead of raising `OutOfBoundsDatetime` (GH14187)
- Bug in `period_range()` ignoring the frequency of `start` and `end` when those are provided as `Period` objects (GH20535).
- Bug in `PeriodIndex` with attribute `freq.n` greater than 1 where adding a `DateOffset` object would return incorrect results (GH23215)
- Bug in `Series` that interpreted string indices as lists of characters when setting datetimelike values (GH23451)
- Bug in `DataFrame` when creating a new column from an ndarray of `Timestamp` objects with timezones creating an object-dtype column, rather than datetime with timezone (GH23932)
- Bug in `Timestamp` constructor which would drop the frequency of an input `Timestamp` (GH22311)
- Bug in `DatetimeIndex` where calling `np.array(dtindex, dtype=object)` would incorrectly return an array of `long` objects (GH23524)
- Bug in `Index` where passing a timezone-aware `DatetimeIndex` and *dtype=object* would incorrectly raise a `ValueError` (GH23524)
- Bug in `Index` where calling `np.array(dtindex, dtype=object)` on a timezone-naive `DatetimeIndex` would return an array of `datetime` objects instead of `Timestamp` objects, potentially losing nanosecond portions of the timestamps (GH23524)
- Bug in `Categorical.__setitem__` not allowing setting with another `Categorical` when both are unordered and have the same categories, but in a different order (GH24142)
- Bug in `date_range()` where using dates with millisecond resolution or higher could return incorrect values or the wrong number of values in the index (GH24110)
- Bug in `DatetimeIndex` where constructing a `DatetimeIndex` from a `Categorical` or `CategoricalIndex` would incorrectly drop timezone information (GH18664)
- Bug in `DatetimeIndex` and `TimedeltaIndex` where indexing with `Ellipsis` would incorrectly lose the index's `freq` attribute (GH21282)
- Clarified error message produced when passing an incorrect `freq` argument to `DatetimeIndex` with `NaT` as the first entry in the passed data (GH11587)
- Bug in `to_datetime()` where `box` and `utc` arguments were ignored when passing a `DataFrame` or `dict` of unit mappings (GH23760)
- Bug in `Series.dt` where the cache would not update properly after an in-place operation (GH24408)
- Bug in `PeriodIndex` where comparisons against an array-like object with length 1 failed to raise `ValueError` (GH23078)
- Bug in `DatetimeIndex.astype()`, `PeriodIndex.astype()` and `TimedeltaIndex.astype()` ignoring the sign of the `dtype` for unsigned integer dtypes (GH24405).
- Fixed bug in `Series.max()` with `datetime64[ns]`-dtype failing to return `NaT` when nulls are present and `skipna=False` is passed (GH24265)
- Bug in `to_datetime()` where arrays of `datetime` objects containing both timezone-aware and timezone-naive `datetimes` would fail to raise `ValueError` (GH24569)
- Bug in `to_datetime()` with invalid datetime format doesn't coerce input to `NaT` even if `errors='coerce'` (GH24763)

Timedelta

- Bug in *DataFrame* with timedelta64[ns] dtype division by Timedelta-like scalar incorrectly returning timedelta64[ns] dtype instead of float64 dtype (GH20088, GH22163)
- Bug in adding a *Index* with object dtype to a *Series* with timedelta64[ns] dtype incorrectly raising (GH22390)
- Bug in multiplying a *Series* with numeric dtype against a timedelta object (GH22390)
- Bug in *Series* with numeric dtype when adding or subtracting an an array or Series with timedelta64 dtype (GH22390)
- Bug in *Index* with numeric dtype when multiplying or dividing an array with dtype timedelta64 (GH22390)
- Bug in *TimedeltaIndex* incorrectly allowing indexing with Timestamp object (GH20464)
- Fixed bug where subtracting *Timedelta* from an object-dtyped array would raise TypeError (GH21980)
- Fixed bug in adding a *DataFrame* with all-*timedelta64[ns]* dtypes to a *DataFrame* with all-integer dtypes returning incorrect results instead of raising TypeError (GH22696)
- Bug in *TimedeltaIndex* where adding a timezone-aware datetime scalar incorrectly returned a timezone-naive *DatetimeIndex* (GH23215)
- Bug in *TimedeltaIndex* where adding np.timedelta64('NaT') incorrectly returned an all-NaT *DatetimeIndex* instead of an all-NaT *TimedeltaIndex* (GH23215)
- Bug in *Timedelta* and *to_timedelta()* have inconsistencies in supported unit string (GH21762)
- Bug in *TimedeltaIndex* division where dividing by another *TimedeltaIndex* raised TypeError instead of returning a *Float64Index* (GH23829, GH22631)
- Bug in *TimedeltaIndex* comparison operations where comparing against non-Timedelta-like objects would raise TypeError instead of returning all-False for __eq__ and all-True for __ne__ (GH24056)
- Bug in *Timedelta* comparisons when comparing with a Tick object incorrectly raising TypeError (GH24710)

Timezones

- Bug in *Index.shift()* where an AssertionError would raise when shifting across DST (GH8616)
- Bug in *Timestamp* constructor where passing an invalid timezone offset designator (Z) would not raise a ValueError (GH8910)
- Bug in *Timestamp.replace()* where replacing at a DST boundary would retain an incorrect offset (GH7825)
- Bug in *Series.replace()* with datetime64[ns, tz] data when replacing NaT (GH11792)
- Bug in *Timestamp* when passing different string date formats with a timezone offset would produce different timezone offsets (GH12064)
- Bug when comparing a tz-naive *Timestamp* to a tz-aware *DatetimeIndex* which would coerce the *DatetimeIndex* to tz-naive (GH12601)
- Bug in *Series.truncate()* with a tz-aware *DatetimeIndex* which would cause a core dump (GH9243)
- Bug in *Series* constructor which would coerce tz-aware and tz-naive *Timestamp* to tz-aware (GH13051)
- Bug in *Index* with datetime64[ns, tz] dtype that did not localize integer data correctly (GH20964)

- Bug in *DatetimeIndex* where constructing with an integer and tz would not localize correctly (GH12619)
- Fixed bug where *DataFrame.describe()* and *Series.describe()* on tz-aware datetimes did not show *first* and *last* result (GH21328)
- Bug in *DatetimeIndex* comparisons failing to raise `TypeError` when comparing timezone-aware `DatetimeIndex` against `np.datetime64` (GH22074)
- Bug in `DataFrame` assignment with a timezone-aware scalar (GH19843)
- Bug in *DataFrame.asof()* that raised a `TypeError` when attempting to compare tz-naive and tz-aware timestamps (GH21194)
- Bug when constructing a *DatetimeIndex* with *Timestamp* constructed with the `replace` method across DST (GH18785)
- Bug when setting a new value with *DataFrame.loc()* with a *DatetimeIndex* with a DST transition (GH18308, GH20724)
- Bug in *Index.unique()* that did not re-localize tz-aware dates correctly (GH21737)
- Bug when indexing a *Series* with a DST transition (GH21846)
- Bug in *DataFrame.resample()* and *Series.resample()* where an `AmbiguousTimeError` or `NonExistentTimeError` would raise if a timezone aware timeseries ended on a DST transition (GH19375, GH10117)
- Bug in *DataFrame.drop()* and *Series.drop()* when specifying a tz-aware Timestamp key to drop from a *DatetimeIndex* with a DST transition (GH21761)
- Bug in *DatetimeIndex* constructor where `NaT` and `dateutil.tz.tzlocal` would raise an `OutOfBoundsDatetime` error (GH23807)
- Bug in *DatetimeIndex.tz_localize()* and *Timestamp.tz_localize()* with `dateutil.tz.tzlocal` near a DST transition that would return an incorrectly localized datetime (GH23807)
- Bug in *Timestamp* constructor where a `dateutil.tz.tzutc` timezone passed with a `datetime.datetime` argument would be converted to a `pytz.UTC` timezone (GH23807)
- Bug in *to_datetime()* where `utc=True` was not respected when specifying a `unit` and `errors='ignore'` (GH23758)
- Bug in *to_datetime()* where `utc=True` was not respected when passing a *Timestamp* (GH24415)
- Bug in *DataFrame.any()* returns wrong value when `axis=1` and the data is of datetimelike type (GH23070)
- Bug in *DatetimeIndex.to_period()* where a timezone aware index was converted to UTC first before creating *PeriodIndex* (GH22905)
- Bug in *DataFrame.tz_localize()*, *DataFrame.tz_convert()*, *Series.tz_localize()*, and *Series.tz_convert()* where `copy=False` would mutate the original argument inplace (GH6326)
- Bug in *DataFrame.max()* and *DataFrame.min()* with `axis=1` where a *Series* with `NaN` would be returned when all columns contained the same timezone (GH10390)

Offsets

- Bug in `FY5253` where date offsets could incorrectly raise an `AssertionError` in arithmetic operations (GH14774)
- Bug in `DateOffset` where keyword arguments `week` and `milliseconds` were accepted and ignored. Passing these will now raise `ValueError` (GH19398)
- Bug in adding `DateOffset` with *`DataFrame`* or *`PeriodIndex`* incorrectly raising `TypeError` (GH23215)
- Bug in comparing `DateOffset` objects with non-DateOffset objects, particularly strings, raising `ValueError` instead of returning `False` for equality checks and `True` for not-equal checks (GH23524)

Numeric

- Bug in *`Series`* `__rmatmul__` doesn't support matrix vector multiplication (GH21530)
- Bug in *`factorize()`* fails with read-only array (GH12813)
- Fixed bug in *`unique()`* handled signed zeros inconsistently: for some inputs 0.0 and -0.0 were treated as equal and for some inputs as different. Now they are treated as equal for all inputs (GH21866)
- Bug in *`DataFrame.agg()`*, *`DataFrame.transform()`* and *`DataFrame.apply()`* where, when supplied with a list of functions and `axis=1` (e.g. `df.apply(['sum', 'mean'], axis=1)`), a `TypeError` was wrongly raised. For all three methods such calculation are now done correctly. (GH16679).
- Bug in *`Series`* comparison against datetime-like scalars and arrays (GH22074)
- Bug in *`DataFrame`* multiplication between boolean dtype and integer returning `object` dtype instead of integer dtype (GH22047, GH22163)
- Bug in *`DataFrame.apply()`* where, when supplied with a string argument and additional positional or keyword arguments (e.g. `df.apply('sum', min_count=1)`), a `TypeError` was wrongly raised (GH22376)
- Bug in *`DataFrame.astype()`* to extension dtype may raise `AttributeError` (GH22578)
- Bug in *`DataFrame`* with `timedelta64[ns]` dtype arithmetic operations with `ndarray` with integer dtype incorrectly treating the narray as `timedelta64[ns]` dtype (GH23114)
- Bug in *`Series.rpow()`* with object dtype `NaN` for `1 ** NA` instead of `1` (GH22922).
- *`Series.agg()`* can now handle numpy NaN-aware methods like `numpy.nansum()` (GH19629)
- Bug in *`Series.rank()`* and *`DataFrame.rank()`* when `pct=True` and more than 2^{24} rows are present resulted in percentages greater than 1.0 (GH18271)
- Calls such as *`DataFrame.round()`* with a non-unique *`CategoricalIndex()`* now return expected data. Previously, data would be improperly duplicated (GH21809).
- Added `log10`, *floor* and *ceil* to the list of supported functions in *`DataFrame.eval()`* (GH24139, GH24353)
- Logical operations `&, |, ^` between *`Series`* and *`Index`* will no longer raise `ValueError` (GH22092)
- Checking PEP 3141 numbers in *`is_scalar()`* function returns `True` (GH22903)
- Reduction methods like *`Series.sum()`* now accept the default value of `keepdims=False` when called from a NumPy ufunc, rather than raising a `TypeError`. Full support for `keepdims` has not been implemented (GH24356).

Conversion

- Bug in *DataFrame.combine_first()* in which column types were unexpectedly converted to float (GH20699)
- Bug in *DataFrame.clip()* in which column types are not preserved and casted to float (GH24162)
- Bug in *DataFrame.clip()* when order of columns of dataframes doesn't match, result observed is wrong in numeric values (GH20911)
- Bug in *DataFrame.astype()* where converting to an extension dtype when duplicate column names are present causes a `RecursionError` (GH24704)

Strings

- Bug in `Index.str.partition()` was not nan-safe (GH23558).
- Bug in `Index.str.split()` was not nan-safe (GH23677).
- Bug *Series.str.contains()* not respecting the `na` argument for a `Categorical` dtype `Series` (GH22158)
- Bug in `Index.str.cat()` when the result contained only `NaN` (GH24044)

Interval

- Bug in the *IntervalIndex* constructor where the `closed` parameter did not always override the inferred `closed` (GH19370)
- Bug in the `IntervalIndex` repr where a trailing comma was missing after the list of intervals (GH20611)
- Bug in *Interval* where scalar arithmetic operations did not retain the `closed` value (GH22313)
- Bug in *IntervalIndex* where indexing with datetime-like values raised a `KeyError` (GH20636)
- Bug in `IntervalTree` where data containing `NaN` triggered a warning and resulted in incorrect indexing queries with *IntervalIndex* (GH23352)

Indexing

- Bug in *DataFrame.ne()* fails if columns contain column name "dtype" (GH22383)
- The traceback from a `KeyError` when asking `.loc` for a single missing label is now shorter and more clear (GH21557)
- *PeriodIndex* now emits a `KeyError` when a malformed string is looked up, which is consistent with the behavior of *DatetimeIndex* (GH22803)
- When `.ix` is asked for a missing integer label in a *MultiIndex* with a first level of integer type, it now raises a `KeyError`, consistently with the case of a flat *Int64Index*, rather than falling back to positional indexing (GH21593)
- Bug in *Index.reindex()* when reindexing a tz-naive and tz-aware *DatetimeIndex* (GH8306)
- Bug in *Series.reindex()* when reindexing an empty series with a `datetime64[ns, tz]` dtype (GH20869)
- Bug in *DataFrame* when setting values with `.loc` and a timezone aware *DatetimeIndex* (GH11365)

- `DataFrame.__getitem__` now accepts dictionaries and dictionary keys as list-likes of labels, consistently with `Series.__getitem__` (GH21294)
- Fixed `DataFrame[np.nan]` when columns are non-unique (GH21428)
- Bug when indexing *`DatetimeIndex`* with nanosecond resolution dates and timezones (GH11679)
- Bug where indexing with a Numpy array containing negative values would mutate the indexer (GH21867)
- Bug where mixed indexes wouldn't allow integers for `.at` (GH19860)
- `Float64Index.get_loc` now raises `KeyError` when boolean key passed. (GH19087)
- Bug in *`DataFrame.loc()`* when indexing with an *`IntervalIndex`* (GH19977)
- *`Index`* no longer mangles `None`, `NaN` and `NaT`, i.e. they are treated as three different keys. However, for numeric Index all three are still coerced to a `NaN` (GH22332)
- Bug in `scalar in Index` if scalar is a float while the `Index` is of integer dtype (GH22085)
- Bug in *`MultiIndex.set_levels()`* when levels value is not subscriptable (GH23273)
- Bug where setting a timedelta column by `Index` causes it to be casted to double, and therefore lose precision (GH23511)
- Bug in *`Index.union()`* and *`Index.intersection()`* where name of the `Index` of the result was not computed correctly for certain cases (GH9943, GH9862)
- Bug in *`Index`* slicing with boolean *`Index`* may raise `TypeError` (GH22533)
- Bug in `PeriodArray.__setitem__` when accepting slice and list-like value (GH23978)
- Bug in *`DatetimeIndex`*, *`TimedeltaIndex`* where indexing with `Ellipsis` would lose their `freq` attribute (GH21282)
- Bug in `iat` where using it to assign an incompatible value would create a new column (GH23236)

Missing

- Bug in *`DataFrame.fillna()`* where a `ValueError` would raise when one column contained a `datetime64[ns, tz]` dtype (GH15522)
- Bug in *`Series.hasnans()`* that could be incorrectly cached and return incorrect answers if null elements are introduced after an initial call (GH19700)
- *`Series.isin()`* now treats all NaN-floats as equal also for `np.object`-dtype. This behavior is consistent with the behavior for float64 (GH22119)
- *`unique()`* no longer mangles NaN-floats and the `NaT`-object for `np.object`-dtype, i.e. `NaT` is no longer coerced to a NaN-value and is treated as a different entity. (GH22295)
- *`DataFrame`* and *`Series`* now properly handle numpy masked arrays with hardened masks. Previously, constructing a DataFrame or Series from a masked array with a hard mask would create a pandas object containing the underlying value, rather than the expected NaN. (GH24574)
- Bug in *`DataFrame`* constructor where `dtype` argument was not honored when handling numpy masked record arrays. (GH24874)

MultiIndex

- Bug in `io.formats.style.Styler.applymap()` where `subset=` with `MultiIndex` slice would reduce to `Series` (GH19861)
- Removed compatibility for `MultiIndex` pickles prior to version 0.8.0; compatibility with `MultiIndex` pickles from version 0.13 forward is maintained (GH21654)
- `MultiIndex.get_loc_level()` (and as a consequence, `.loc` on a `Series` or `DataFrame` with a `MultiIndex` index) will now raise a `KeyError`, rather than returning an empty `slice`, if asked a label which is present in the `levels` but is unused (GH22221)
- `MultiIndex` has gained the `MultiIndex.from_frame()`, it allows constructing a `MultiIndex` object from a `DataFrame` (GH22420)
- Fix `TypeError` in Python 3 when creating `MultiIndex` in which some levels have mixed types, e.g. when some labels are tuples (GH15457)

I/O

- Bug in `read_csv()` in which a column specified with `CategoricalDtype` of boolean categories was not being correctly coerced from string values to booleans (GH20498)
- Bug in `read_csv()` in which unicode column names were not being properly recognized with Python 2.x (GH13253)
- Bug in `DataFrame.to_sql()` when writing timezone aware data (`datetime64[ns, tz]` dtype) would raise a `TypeError` (GH9086)
- Bug in `DataFrame.to_sql()` where a naive `DatetimeIndex` would be written as `TIMESTAMP WITH TIMEZONE` type in supported databases, e.g. PostgreSQL (GH23510)
- Bug in `read_excel()` when `parse_cols` is specified with an empty dataset (GH9208)
- `read_html()` no longer ignores all-whitespace `<tr>` within `<thead>` when considering the `skiprows` and `header` arguments. Previously, users had to decrease their `header` and `skiprows` values on such tables to work around the issue. (GH21641)
- `read_excel()` will correctly show the deprecation warning for previously deprecated `sheetname` (GH17994)
- `read_csv()` and `read_table()` will throw `UnicodeError` and not coredump on badly encoded strings (GH22748)
- `read_csv()` will correctly parse timezone-aware datetimes (GH22256)
- Bug in `read_csv()` in which memory management was prematurely optimized for the C engine when the data was being read in chunks (GH23509)
- Bug in `read_csv()` in unnamed columns were being improperly identified when extracting a multi-index (GH23687)
- `read_sas()` will parse numbers in sas7bdat-files that have width less than 8 bytes correctly. (GH21616)
- `read_sas()` will correctly parse sas7bdat files with many columns (GH22628)
- `read_sas()` will correctly parse sas7bdat files with data page types having also bit 7 set (so page type is 128 + 256 = 384) (GH16615)
- Bug in `read_sas()` in which an incorrect error was raised on an invalid file format. (GH24548)

- Bug in `detect_client_encoding()` where potential `IOError` goes unhandled when importing in a mod_wsgi process due to restricted access to stdout. (GH21552)
- Bug in `DataFrame.to_html()` with `index=False` misses truncation indicators (...) on truncated DataFrame (GH15019, GH22783)
- Bug in `DataFrame.to_html()` with `index=False` when both columns and row index are `MultiIndex` (GH22579)
- Bug in `DataFrame.to_html()` with `index_names=False` displaying index name (GH22747)
- Bug in `DataFrame.to_html()` with `header=False` not displaying row index names (GH23788)
- Bug in `DataFrame.to_html()` with `sparsify=False` that caused it to raise `TypeError` (GH22887)
- Bug in `DataFrame.to_string()` that broke column alignment when `index=False` and width of first column's values is greater than the width of first column's header (GH16839, GH13032)
- Bug in `DataFrame.to_string()` that caused representations of `DataFrame` to not take up the whole window (GH22984)
- Bug in `DataFrame.to_csv()` where a single level MultiIndex incorrectly wrote a tuple. Now just the value of the index is written (GH19589).
- `HDFStore` will raise `ValueError` when the `format` kwarg is passed to the constructor (GH13291)
- Bug in `HDFStore.append()` when appending a `DataFrame` with an empty string column and `min_itemsize` < 8 (GH12242)
- Bug in `read_csv()` in which memory leaks occurred in the C engine when parsing `NaN` values due to insufficient cleanup on completion or error (GH21353)
- Bug in `read_csv()` in which incorrect error messages were being raised when `skipfooter` was passed in along with `nrows`, `iterator`, or `chunksize` (GH23711)
- Bug in `read_csv()` in which `MultiIndex` index names were being improperly handled in the cases when they were not provided (GH23484)
- Bug in `read_csv()` in which unnecessary warnings were being raised when the dialect's values conflicted with the default arguments (GH23761)
- Bug in `read_html()` in which the error message was not displaying the valid flavors when an invalid one was provided (GH23549)
- Bug in `read_excel()` in which extraneous header names were extracted, even though none were specified (GH11733)
- Bug in `read_excel()` in which column names were not being properly converted to string sometimes in Python 2.x (GH23874)
- Bug in `read_excel()` in which `index_col=None` was not being respected and parsing index columns anyway (GH18792, GH20480)
- Bug in `read_excel()` in which `usecols` was not being validated for proper column names when passed in as a string (GH20480)
- Bug in `DataFrame.to_dict()` when the resulting dict contains non-Python scalars in the case of numeric data (GH23753)
- `DataFrame.to_string()`, `DataFrame.to_html()`, `DataFrame.to_latex()` will correctly format output when a string is passed as the `float_format` argument (GH21625, GH22270)
- Bug in `read_csv()` that caused it to raise `OverflowError` when trying to use 'inf' as `na_value` with integer index column (GH17128)

- Bug in *read_csv()* that caused the C engine on Python 3.6+ on Windows to improperly read CSV filenames with accented or special characters (GH15086)
- Bug in *read_fwf()* in which the compression type of a file was not being properly inferred (GH22199)
- Bug in `pandas.io.json.json_normalize()` that caused it to raise `TypeError` when two consecutive elements of `record_path` are dicts (GH22706)
- Bug in *DataFrame.to_stata()*, `pandas.io.stata.StataWriter` and `pandas.io.stata.StataWriter117` where a exception would leave a partially written and invalid dta file (GH23573)
- Bug in *DataFrame.to_stata()* and `pandas.io.stata.StataWriter117` that produced invalid files when using strLs with non-ASCII characters (GH23573)
- Bug in `HDFStore` that caused it to raise `ValueError` when reading a Dataframe in Python 3 from fixed format written in Python 2 (GH24510)
- Bug in *DataFrame.to_string()* and more generally in the floating `repr` formatter. Zeros were not trimmed if `inf` was present in a columns while it was the case with NA values. Zeros are now trimmed as in the presence of NA (GH24861).
- Bug in the `repr` when truncating the number of columns and having a wide last column (GH24849).

Plotting

- Bug in *DataFrame.plot.scatter()* and *DataFrame.plot.hexbin()* caused x-axis label and ticklabels to disappear when colorbar was on in IPython inline backend (GH10611, GH10678, and GH20455)
- Bug in plotting a Series with datetimes using `matplotlib.axes.Axes.scatter()` (GH22039)
- Bug in *DataFrame.plot.bar()* caused bars to use multiple colors instead of a single one (GH20585)
- Bug in validating color parameter caused extra color to be appended to the given color array. This happened to multiple plotting functions using matplotlib. (GH20726)

Groupby/resample/rolling

- Bug in `pandas.core.window.Rolling.min()` and `pandas.core.window.Rolling.max()` with `closed='left'`, a datetime-like index and only one entry in the series leading to segfault (GH24718)
- Bug in *pandas.core.groupby.GroupBy.first()* and *pandas.core.groupby.GroupBy.last()* with `as_index=False` leading to the loss of timezone information (GH15884)
- Bug in `DateFrame.resample()` when downsampling across a DST boundary (GH8531)
- Bug in date anchoring for `DateFrame.resample()` with offset `Day` when n > 1 (GH24127)
- Bug where `ValueError` is wrongly raised when calling `count()` method of a `SeriesGroupBy` when the grouping variable only contains NaNs and numpy version < 1.13 (GH21956).
- Multiple bugs in `pandas.core.window.Rolling.min()` with `closed='left'` and a datetime-like index leading to incorrect results and also segfault. (GH21704)
- Bug in *pandas.core.resample.Resampler.apply()* when passing positional arguments to applied func (GH14615).
- Bug in *Series.resample()* when passing `numpy.timedelta64` to `loffset` kwarg (GH7687).
- Bug in *pandas.core.resample.Resampler.asfreq()* when frequency of `TimedeltaIndex` is a subperiod of a new frequency (GH13022).

- Bug in `pandas.core.groupby.SeriesGroupBy.mean()` when values were integral but could not fit inside of int64, overflowing instead. (GH22487)
- `pandas.core.groupby.RollingGroupby.agg()` and `pandas.core.groupby.ExpandingGroupby.agg()` now support multiple aggregation functions as parameters (GH15072)
- Bug in `DataFrame.resample()` and `Series.resample()` when resampling by a weekly offset (`'W'`) across a DST transition (GH9119, GH21459)
- Bug in `DataFrame.expanding()` in which the `axis` argument was not being respected during aggregations (GH23372)
- Bug in `pandas.core.groupby.GroupBy.transform()` which caused missing values when the input function can accept a `DataFrame` but renames it (GH23455).
- Bug in `pandas.core.groupby.GroupBy.nth()` where column order was not always preserved (GH20760)
- Bug in `pandas.core.groupby.GroupBy.rank()` with `method='dense'` and `pct=True` when a group has only one member would raise a `ZeroDivisionError` (GH23666).
- Calling `pandas.core.groupby.GroupBy.rank()` with empty groups and `pct=True` was raising a `ZeroDivisionError` (GH22519)
- Bug in `DataFrame.resample()` when resampling `NaT` in `TimeDeltaIndex` (GH13223).
- Bug in `DataFrame.groupby()` did not respect the `observed` argument when selecting a column and instead always used `observed=False` (GH23970)
- Bug in `pandas.core.groupby.SeriesGroupBy.pct_change()` or `pandas.core.groupby.DataFrameGroupBy.pct_change()` would previously work across groups when calculating the percent change, where it now correctly works per group (GH21200, GH21235).
- Bug preventing hash table creation with very large number (2^32) of rows (GH22805)
- Bug in groupby when grouping on categorical causes `ValueError` and incorrect grouping if `observed=True` and `nan` is present in categorical column (GH24740, GH21151).

Reshaping

- Bug in `pandas.concat()` when joining resampled DataFrames with timezone aware index (GH13783)
- Bug in `pandas.concat()` when joining only *Series* the *names* argument of *concat* is no longer ignored (GH23490)
- Bug in `Series.combine_first()` with `datetime64[ns, tz]` dtype which would return tz-naive result (GH21469)
- Bug in `Series.where()` and `DataFrame.where()` with `datetime64[ns, tz]` dtype (GH21546)
- Bug in `DataFrame.where()` with an empty DataFrame and empty `cond` having non-bool dtype (GH21947)
- Bug in `Series.mask()` and `DataFrame.mask()` with `list` conditionals (GH21891)
- Bug in `DataFrame.replace()` raises RecursionError when converting OutOfBounds `datetime64[ns, tz]` (GH20380)
- `pandas.core.groupby.GroupBy.rank()` now raises a `ValueError` when an invalid value is passed for argument `na_option` (GH22124)
- Bug in `get_dummies()` with Unicode attributes in Python 2 (GH22084)
- Bug in `DataFrame.replace()` raises `RecursionError` when replacing empty lists (GH22083)

- Bug in `Series.replace()` and `DataFrame.replace()` when dict is used as the `to_replace` value and one key in the dict is is another key's value, the results were inconsistent between using integer key and using string key (GH20656)
- Bug in `DataFrame.drop_duplicates()` for empty `DataFrame` which incorrectly raises an error (GH20516)
- Bug in `pandas.wide_to_long()` when a string is passed to the stubnames argument and a column name is a substring of that stubname (GH22468)
- Bug in `merge()` when merging `datetime64[ns, tz]` data that contained a DST transition (GH18885)
- Bug in `merge_asof()` when merging on float values within defined tolerance (GH22981)
- Bug in `pandas.concat()` when concatenating a multicolumn DataFrame with tz-aware data against a DataFrame with a different number of columns (GH22796)
- Bug in `merge_asof()` where confusing error message raised when attempting to merge with missing values (GH23189)
- Bug in `DataFrame.nsmallest()` and `DataFrame.nlargest()` for dataframes that have a `MultiIndex` for columns (GH23033).
- Bug in `pandas.melt()` when passing column names that are not present in `DataFrame` (GH23575)
- Bug in `DataFrame.append()` with a `Series` with a dateutil timezone would raise a `TypeError` (GH23682)
- Bug in `Series` construction when passing no data and `dtype=str` (GH22477)
- Bug in `cut()` with `bins` as an overlapping `IntervalIndex` where multiple bins were returned per item instead of raising a `ValueError` (GH23980)
- Bug in `pandas.concat()` when joining `Series` datetimetz with `Series` category would lose timezone (GH23816)
- Bug in `DataFrame.join()` when joining on partial MultiIndex would drop names (GH20452).
- `DataFrame.nlargest()` and `DataFrame.nsmallest()` now returns the correct n values when keep != 'all' also when tied on the first columns (GH22752)
- Constructing a DataFrame with an index argument that wasn't already an instance of `Index` was broken (GH22227).
- Bug in `DataFrame` prevented list subclasses to be used to construction (GH21226)
- Bug in `DataFrame.unstack()` and `DataFrame.pivot_table()` returning a missleading error message when the resulting DataFrame has more elements than int32 can handle. Now, the error message is improved, pointing towards the actual problem (GH20601)
- Bug in `DataFrame.unstack()` where a `ValueError` was raised when unstacking timezone aware values (GH18338)
- Bug in `DataFrame.stack()` where timezone aware values were converted to timezone naive values (GH19420)
- Bug in `merge_asof()` where a `TypeError` was raised when `by_col` were timezone aware values (GH21184)
- Bug showing an incorrect shape when throwing error during `DataFrame` construction. (GH20742)

Sparse

- Updating a boolean, datetime, or timedelta column to be Sparse now works (GH22367)
- Bug in `Series.to_sparse()` with Series already holding sparse data not constructing properly (GH22389)
- Providing a `sparse_index` to the SparseArray constructor no longer defaults the na-value to `np.nan` for all dtypes. The correct na_value for `data.dtype` is now used.
- Bug in `SparseArray.nbytes` under-reporting its memory usage by not including the size of its sparse index.
- Improved performance of *`Series.shift()`* for non-NA `fill_value`, as values are no longer converted to a dense array.
- Bug in `DataFrame.groupby` not including `fill_value` in the groups for non-NA `fill_value` when grouping by a sparse column (GH5078)
- Bug in unary inversion operator (`~`) on a `SparseSeries` with boolean values. The performance of this has also been improved (GH22835)
- Bug in `SparseArary.unique()` not returning the unique values (GH19595)
- Bug in `SparseArray.nonzero()` and `SparseDataFrame.dropna()` returning shifted/incorrect results (GH21172)
- Bug in *`DataFrame.apply()`* where dtypes would lose sparseness (GH23744)
- Bug in *`concat()`* when concatenating a list of *`Series`* with all-sparse values changing the `fill_value` and converting to a dense Series (GH24371)

Style

- *`background_gradient()`* now takes a `text_color_threshold` parameter to automatically lighten the text color based on the luminance of the background color. This improves readability with dark background colors without the need to limit the background colormap range. (GH21258)
- *`background_gradient()`* now also supports tablewise application (in addition to rowwise and columnwise) with `axis=None` (GH15204)
- *`bar()`* now also supports tablewise application (in addition to rowwise and columnwise) with `axis=None` and setting clipping range with `vmin` and `vmax` (GH21548 and GH21526). `NaN` values are also handled properly.

Build changes

- Building pandas for development now requires `cython >= 0.28.2` (GH21688)
- Testing pandas now requires `hypothesis>=3.58`. You can find the Hypothesis docs here, and a pandas-specific introduction *in the contributing guide*. (GH22280)
- Building pandas on macOS now targets minimum macOS 10.9 if run on macOS 10.9 or above (GH23424)

Other

- Bug where C variables were declared with external linkage causing import errors if certain other C libraries were imported before Pandas. (GH24113)

Contributors

A total of 337 people contributed patches to this release. People with a "+" by their names contributed a patch for the first time.

- AJ Dyka +
- AJ Pryor, Ph.D +
- Aaron Critchley
- Adam Hooper
- Adam J. Stewart
- Adam Kim
- Adam Klimont +
- Addison Lynch +
- Alan Hogue +
- Alex Radu +
- Alex Rychyk
- Alex Strick van Linschoten +
- Alex Volkov +
- Alexander Buchkovsky
- Alexander Hess +
- Alexander Ponomaroff +
- Allison Browne +
- Aly Sivji
- Andrew
- Andrew Gross +
- Andrew Spott +
- Andy +
- Aniket uttam +
- Anjali2019 +
- Anjana S +
- Antti Kaihola +
- Anudeep Tubati +
- Arjun Sharma +
- Armin Varshokar

- Artem Bogachev
- ArtinSarraf +
- Barry Fitzgerald +
- Bart Aelterman +
- Ben James +
- Ben Nelson +
- Benjamin Grove +
- Benjamin Rowell +
- Benoit Paquet +
- Boris Lau +
- Brett Naul
- Brian Choi +
- C.A.M. Gerlach +
- Carl Johan +
- Chalmer Lowe
- Chang She
- Charles David +
- Cheuk Ting Ho
- Chris
- Chris Roberts +
- Christopher Whelan
- Chu Qing Hao +
- Da Cheezy Mobsta +
- Damini Satya
- Daniel Himmelstein
- Daniel Saxton +
- Darcy Meyer +
- DataOmbudsman
- David Arcos
- David Krych
- Dean Langsam +
- Diego Argueta +
- Diego Torres +
- Dobatymo +
- Doug Latornell +
- Dr. Irv