CSCD01 Deliverable 2

Team 12

Rahmatullah Nikyar, Frank Xu, Usman Siddiqui, Weiqiang Zhang, Tony Zeng

Detailed description of each bug/feature request

Issue #: **31220**
Description:
Index.sort_values places NaN values at the beginning of the array. This differs from Series.sort_values which places them at the end of the array. The expected behavior is that they should sort the values in a consistent way.
Example:

```
>>> pd.Index([1, np.nan, 0]).sort_values(ascending=False)
Float64Index([nan, 1.0, 0.0], dtype='float64')
>>> pd.Series([1, np.nan, 0]).sort_values(ascending=False)
0    1.0
2    0.0
1    NaN
dtype: float64
```

Estimate: 2 hours

Issue #: **28556**
Description:
When reading a string numeric value from a json, pd.read_json does not maintain the string nature of the value.
Example:

```
>>> df = pd.DataFrame(range(3), index=list("123"))
>>> df.to_json(orient="split")
'{"columns":[0],"index":["1","2","3"],"data":[[0],[1],[2]]}'
>>> pd.read_json(df.to_json(orient="split"), orient="split").index
Int64Index([1, 2, 3], dtype='int64')
```

Estimate: 3 hours

Issue #: **28572**
Description:
There exist inconsistency with groupby function when providing a single datetime grouper object vs applying a column name and a datetime grouper.
In the case where a datetime column in the target DataFrame is specified and all entries are removed for a specific time category (day, month, year, etc), applying a group by with a column name and a datetime grouper will omit the record for the grouped by time category in the resulting dataframe after an aggregation function. However, when applying just the datetime grouper, indexing a column, and then applying the aggregation function the omitted record for the missing time category will appear.
Example:

```
df = pd.DataFrame({'Dates': dates,
                   'Sales': sales,
                   'Product': product
                  })
```

```
// omit all the sales records in march
>>> march = df.Dates.dt.month == 3
>>> df = df[~march]

// monthly datetime grouper defined
>>> monthly = pd.Grouper(key='Dates', freq='M')
// using groupby function, supplying column and monthly grouper
>>> sum_sales = df.groupby(['Product', monthly])['Sales'].sum()

// notice missing entry for March (03)
Product  Dates
A        2001-01-31    658
         2001-02-28    460
         2001-04-30    541
         2001-05-31    701
         2001-06-30    517
         2001-07-31    596
         2001-08-31    802
         2001-09-30    654
         2001-10-31    561
         2001-11-30    473
         2001-12-31    605

// using only the monthly grouper
df.groupby(monthly)['Sales'].sum()

// notice month of march is displayed this time
Dates
2001-01-31    1616
2001-02-28    1256
2001-03-31       0
2001-04-30    1555
2001-05-31    1384
2001-06-30    1451
2001-07-31    1677
2001-08-31    1472
2001-09-30    1535
2001-10-31    1316
2001-11-30    1573
2001-12-31    1403
```

Estimate: 4 hours

Issue #: **16097**

Description:

Invalid arguments when using the set_option function with ‘max_colwidth’ does not give a warning. The max column width should be no less than 3, this is because Ellipses only take 3 characters. We can resolve this issue by checking the max column size, outputting the warning accordingly, and forbidding negative column width integers.

Example:

```
import pandas as pd

df = pd.DataFrame(np.array([['foo', 'bar', 'bim', 'uncomfortably long string'],
['horse', 'cow', 'banana', 'apple']]))
df
#        0    1       2                          3
# 0    foo  bar     bim  uncomfortably long string
# 1  horse  cow  banana                      apple

pd.set_option('max_colwidth', 2)
df
#        0    1       2                          3
# 0    foo  bar     bim  uncomfortably long string
# 1  horse  cow  banana                      apple

pd.set_option('max_colwidth', 6)
df
#        0    1      2      3
# 0    foo  bar    bim  un...
# 1  horse  cow  ba...  apple
```

Estimate: 3 hours

Issue #: **32326**
Description:
Invalid arguments when using the to_json function does not give a warning. According to documentation, compression parameter is only applicable if a file name is given. If the filename argument is not given, but the compression argument is given, then users should receive a warning.
Example:

```
import pandas as pd

df = pd.DataFrame({"a": [1, 2, 3, 4], "b": ["A", "B", "C", "D"],})
json_df = df.to_json(lines=True, orient="records", compression="gzip")
print(json_df)
```

Estimate: 1 hour

Best Issues to Work on:

[16097]: We picked this specific issue because of its simplicity and relatively high impact for new users. One risk we anticipate is that our implementation has the chance to disturb existing validations for set_option. Because the concept of the implementation is relatively easy to understand, the majority of the estimated effort will go towards designing the validation itself. Most of the time will be allocated for deciding where the best place to put the validation function is so it is easy to modify in the future and does not clash with existing validations. The downside of not having this feature is that the user won't know why inputs 0, 1, 2 for max_colwidth don't actually do anything even though all integer inputs are valid. The problem is not stated in the docs, so for a new Pandas user their only option is to look at the source code, and that could end up wasting a lot of their time on a simple problem.

[32326]: This issue was chosen because it improves usability of Pandas for new users. Improving the user experience is highly important. It reduces complications for a fairly complex library. The estimated effort includes implementation and validation, and it is estimated to be a quick fix because it is a straightforward task. A risk to consider is if the "compression" argument is used in other cases in future patches. We must document our changes so that future programmers can correctly upgrade it.

Technical Commentary

[16097]: We added a warning for the maximum width column size if the user set the option to less than 4. This does not have a great effect on the design of the project. Rather, it covers a flaw in the design by warning the user of potential issues of an action. This improves the user experience of pandas. The fix was checking for if the maximum column width is set is less than 4. So in config_init.py, on the register option of the call back, we created a wrapper function for better design and called the function that warned if the maximum column width was less than zero and the function that we created to check if it was less than four. This is better design as incase in the future they want to remove the check for less than zero or modify it, it wouldn't affect our code as the other option was putting the check inside the same function.
Files Modified: pandas/core/config/config_init.py

[32326]: Added a warning when calling to_json that warns the user that they are not actually compressing any data. This happens when to_json is supplied a compression parameter but no file path is specified. This change affects the verbosity of saving both dataframe and series object types into json and does not change the code too much. Right after the function checks for file path existence, we insert another check for if compression is specified or not. If compression is not specified we add the warning and then the code continues to return the json object conversion.

Files Modified: pandas/io/json/_json.py

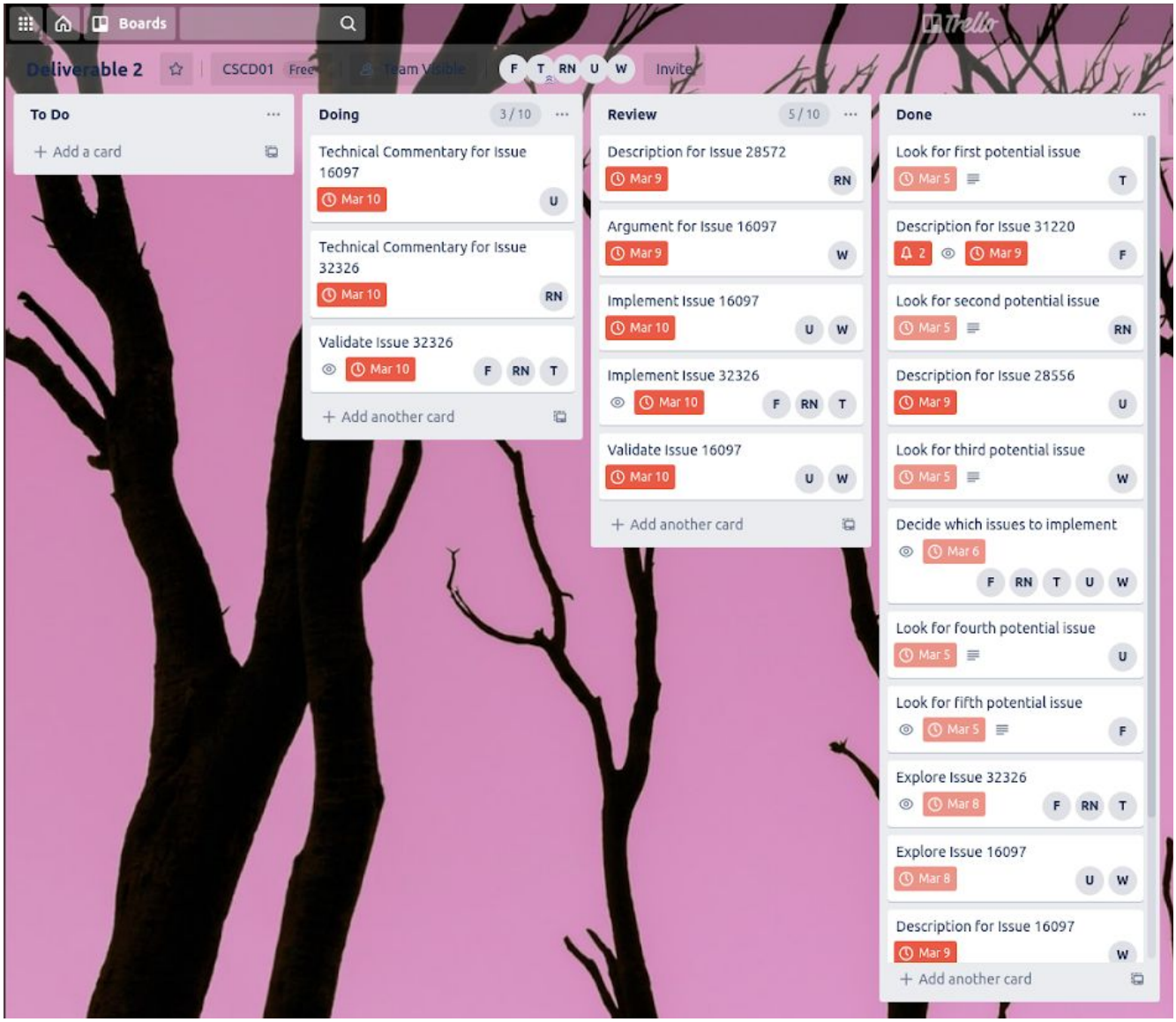
Acceptance Test Explained

Forked repo: https://github.com/R4HMATT/pandas.git

For our issue of 16097, the suite of acceptance tests can be found on branch "warn_set_option_max_colwidth_size" and in pandas/tests/config/test_max_colwidth_warning.py. To test it, run the command "pytest test_max_colwidth_warning.py" in the pandas/tests/config folder. The tests are documented to explain what is being tested.

For our issue of 32326, the suite of acceptance tests can be found in the file of pandas/tests/io/json/test_to_json_compression_warnings.py. Test file is set with 2 test classes, testing Series objects and DataFrame objects to_json. Both contain similar tests. We took advantage of the python Warning module's catch_warning context manager to be able to test warnings that occur. Also used panda's ensure_clean context manager to generate disposable file paths for test purposes. For details on test cases please see comments inside of file.

Software Development Process Followed



We followed the software development process of Kanban. Each member assigned themselves tasks and a WIP was set to 2. This ensured that the ‘Doing’ column was never over cluttered with too many tasks and everything was going as planned. We divided the things that needed to be done for this deliverable into small tasks to keep everything organized and using Trello allowed us to see what each team member was currently working on. Our communication was done through Facebook messenger with a maximum of 2 hour wait time for responses and we were either doing discord calls or meeting in person at IC multiple times weekly.