CSCD01 Deliverable 4

Team Name: Waterboys

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

Chosen Issue 32450:
No current direct conversion between stringDtype series to Inte64Dtype series. The poster of the issue is able to convert to the latter type by indirectly calling the to_numeric method, but there are other issues when converting with the to_numeric method as well. To_numeric requires the unnessary parameter errors='coerce' and needs convert_dtypes to be called afterwards even though it should be returning Int64 Dtype directly. However, there should be a direct way to convert such types. The poster did make a mistake on their first test, where astype(“Int64”) was attempted, this is not supposed to work because Int64 isn’t a dtype of numpy but int64 is. The problem lies in the fact that “astype(‘int64)” is called on a StringArray with panda’s NA type, numpy does not know how to handle converting Series and Dataframe typings when it includes pandas.NA or numpy.nan.

Original Implementation Plan:
After digging through the codecase we discovered that using astype to turn an integer series into a string series will still work with a pandas.NA type in the series. The way pandas has handled the work around to make this work is having a mask object stored where it’s a list of booleans, having True values mirroring the pandas.NA type in the stored ndarray. This fixes the problem because the ndarray can just have integers in place of pandas.NA. IntegerArray inherits from BaseMaskedArray and StringArray just inherits from PandasArray. The high level plan to fix issue 32450 is going to be to make StringArray inherit from BaseMaskedArray and partly, if not fully implement StringArray. Since StringArray is used a lot everywhere in both Series and Dataframe we’re going to have to test thoroughly to make sure we don’t break any of the previous functionalities.

Explanation of why we didn’t use implementation from D3 (Original Plan):
We initially came up with the implementation for D3 because we noticed that IntegerArray, which extended off BaseMaskedArray, was able to properly handle pd.NA conversions. We noticed that aside from fixing the problem at hand, the masked approach to arrays provided a lot of benefits. However, after much more careful analysis of our proposed solution, we realized that the amount of changes required was originally underestimated. It was not reasonable to refactor the StringArray object from inheriting PandasArray to BaseMaskedArray within the allocated time. There are too many dependencies that rely on the PandasArray implementation of StringArray. Frankly, we were not ready to handle such a substantial structural change. Nevertheless, our newly proposed solution, albeit simple, still solves the problem effectively.

The implementation we did:
The block class contains the array values and haa method called astype that takes the numpy array value and calls the numpy internal astype method with it. In said method there is a check for if the value is a pandas extension dtype, which are (string types are), we added an additional check to see if the data type we are converting to is numpy’s int64 or pandas own extension integer type called _IntegerDtype. Following the check, we take the original values of the array and we create a copy of them, but replace every pd.NA entry with 0. Then we create another copy of the original values called the mask, and replaces every pd.NA entry with a boolean

True, and every other value with a False. This line of thinking follows very closely on how the actual IntegerArray is implemented, making use of the mask idea. We use the previous tactic for ObjectArrays, ObjectArrays is used to hold strings before StringArrays were created and can currently still handle strings. We needed to fix the ObjectArray to IntegerArray conversion as well.

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

For our issue of 32450, the suite of unit tests can be found on branch `issue-32450` and in `pandas/tests/config/test_fix_32450.py`.

To test it, run the command "pytest `test_fix_32450.py`" in the `pandas/tests/config/` folder. The tests are documented to explain what is being tested.

Test file has a class with 4 test functions, testing the conversion of StringArray and Object Array to an Integer Array. The tests also include "pd.Na" values and without "pd.Na" values. For details on test cases please see comments inside of the file.

Acceptance Tests for #32450:

```
x = pd.Series(['1', pd.NA, '3'], dtype=pd.StringDtype())
x.astype("int64")
0      1
1   <NA>
2      3
dtype: int64
```

```
x = pd.DataFrame({'a': ['1', '3'], b: [pd.NA, '2']}, dtype=pd.StringDtype())
x.astype(int)
       a      b
0      1   <NA>
1      3      2
```

```
x = pd.Series(['1', pd.NA, '3'], dtype='O')
x.astype(int)
0      1
1   <NA>
2      3
dtype: int64
```

```
x = pd.DataFrame({'a': ['1', '3'], b: [pd.NA, '2']}, dtype='O')
x.astype("int64")
       a      b
0      1   <NA>
1      3      2
```

Sequence Diagram Link:
https://github.com/CSCD01/team_12-project/blob/master/d4/d4_seq_after.png

User Guide:
Series

- Initialize a string dtype Series
- Use astype and convert it to an Integer of your choosing
- Let's say for example:

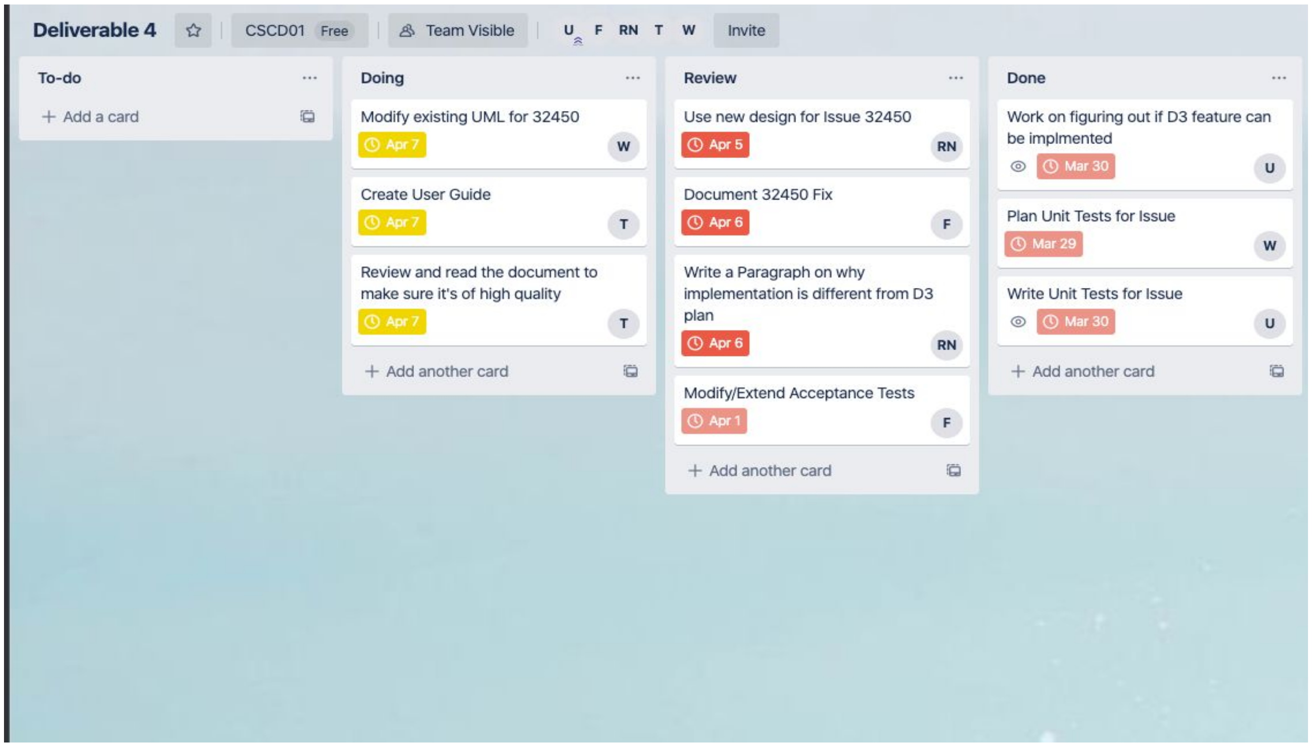
```
x = pd.Series(['1', pd.NA, '3'], dtype=pd.StringDtype())
x.astype(“int64”)
0      1
1   <NA>
2      3
dtype: Int64
```

DataFrame

- Initialize a string dtype DataFrame
- Use astype and convert it to an Integer of your choosing
- Let's say for example:

```
x = pd.DataFrame({‘a’: [‘1’, ‘3’], b: [pd.NA, ‘2’]}, dtype=pd.StringDtype())
x.astype(int)
       a      b
0      1   <NA>
1      3      2
```

Software Development Process Followed:


Deliverable 4 | CSCD01 Free | Team Visible | U F RN T W | Invite

To-do
+ Add a card

Doing
- Modify existing UML for 32450 — Apr 7 — W
- Create User Guide — Apr 7 — T
- Review and read the document to make sure it's of high quality — Apr 7 — T
+ Add another card

Review
- Use new design for Issue 32450 — Apr 5 — RN
- Document 32450 Fix — Apr 6 — F
- Write a Paragraph on why implementation is different from D3 plan — Apr 6 — RN
- Modify/Extend Acceptance Tests — Apr 1 — F
+ Add another card

Done
- Work on figuring out if D3 feature can be implmented — Mar 30 — U
- Plan Unit Tests for Issue — Mar 29 — W
- Write Unit Tests for Issue — Mar 30 — U
+ Add another card

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 started doing all our meetings through discord because of the current pandemic.