import numpy as np  
import pandas as pd  
  
pd.options.display.max\_rows = 999  
pd.options.display.max\_columns = 999  
pd.options.display.max\_colwidth = 150

**Hinata**: Hey, Kageyama, what’s that stuff in the first box?

**Kageyama**: The ‘box’ is called a cell. Didn’t you look at the [Jupyterlab Docs](https://jupyterlab.readthedocs.io/en/stable/user/notebook.html) or [Jupyter Notebook Tutorial](https://www.dataquest.io/blog/jupyter-notebook-tutorial/) Suga recommended?

**Tsukishima**: The Tutorial is very basic. Even you should be able to follow it.

**Hinata**: Hey!!

**Kageyama**: If you can’t sit still long enough for those, here’s [a video](https://www.youtube.com/watch?v=K2Yb1nXTmYM) that’s less than five minutes.

**Tsukishima**: You’re overestimating his intelligence and reasoning skills. Here’s a [two hour video on Jupyterlab basics](https://www.youtube.com/watch?v=Gzun8PpyBCo). Maybe you’ll come away from that a little smarter.

**Sugawara**: Don’t forget, Tsukishima. Not everyone has the computer science background you do.

**Tsukishima**: *Tsk*

**Sugawara**: Kageyama, why don’t you explain the code you put in the first cell.

**Kageyama**: It’s two libraries and some pandas display settings.

**Hinata**: What’re libraries?

**Kageyama**: Learn to code, dumbass! Haven’t you finished the [University of Michigan Python 3 Programming Specialization](https://www.coursera.org/specializations/python-3-programming?) I showed you??

**Hinata**: It’s a set of five courses and each one takes like 20 hours! It’s gonna take *forever*!!!

**Yachi**: Those courses are in English. Shouldn’t we give him ones in Japanese?

**Kageyama**: If he’s gonna be needy, he can find his own courses.

**Tsukishima**: If he ever does manage to finish it, he can say he took a course from a US university. I’m sure his resume could use all the padding it can get.

**Hinata**: C’mon guys, what’s a library?

**Tsukishima**: *Tsk*

**Tsukishima**: A place with a lot of books.

**Kageyama**: *Glare*

**Kageyama**: Libraries are fundamental to high level computer languages.

**Yachi**: A **high level computer language** is one that’s easier for *people* to read and write.

**Yachi**: Computers understand things in 1s and 0s. Languages closer to what *computers* understand are called **low level computer languages** and are generally really hard for people. Examples include assembly or machine code.

**Yachi**: Because they’re easier for people to work with, there are a lot more high level languages. Some of them are Python, R, Ruby, Julia, and Java.

**Kageyama**: Right. Yeah. What she said.

**Kageyama**: So, a library is written in a specific language and is basically a collection of code that’s already written for you.

**Yamaguchi**: Loading every library for a given language would use up a lot of your computer’s memory, so you only load the ones you need. Each library has a group of related functions for specific kinds of tasks.

**Hinata**: *Oooohhh!*

**Hinata**: What about the display settings? What’s the 999 stuff about?

**Yamaguchi**: By default pandas only shows **60 rows and 20 columns**. Kageyama got the biggest dataset, so probably he probably upped the number of rows and columns so he could see more data at once.

**Hinata**: Oh. Okay!

## Load data

keep\_cols = ['node\_id', 'sourceID', 'name', 'incorporation\_date', 'country\_codes', 'countries',  
 'jurisdiction\_description', 'jurisdiction', 'service\_provider', 'status']  
dtypes = {'node\_id': 'int32', 'sourceID':'category', 'name':'object', 'country\_codes':'category', 'countries':'category',  
 'jurisdiction\_description':'category', 'jurisdiction':'category', 'service\_provider':'category', 'status':'category'}  
  
bahamas\_entity\_raw = pd.read\_csv('../data/raw/bahamas\_leaks/bahamas\_leaks.nodes.entity.csv',   
 usecols = keep\_cols,  
 dtype=dtypes)  
offshore\_entity\_raw = pd.read\_csv('../data/raw/offshore\_leaks/offshore\_leaks.nodes.entity.csv',   
 usecols = keep\_cols,  
 dtype=dtypes)  
panama\_entity\_raw = pd.read\_csv('../data/raw/panama\_papers/panama\_papers.nodes.entity.csv',   
 usecols = keep\_cols,  
 dtype=dtypes)  
paradise\_entity\_raw = pd.read\_csv('../data/raw/paradise\_papers/paradise\_papers.nodes.entity.csv',   
 usecols = keep\_cols,  
 dtype=dtypes)

**Hinata**: Wait! Wait! What’re ‘keep\_cols’ and ‘dtypes’? And why’d you put them in the read\_csv function after the file path??

**Tsukishima**: We’re never going to get through this notebook.

**Kageyama**: ‘keep\_cols’ is a list variable representing the columns we want to keep from the original data. The CSVs had a bunch of empty or extraneous columns.

**Kageyama**: ‘dtypes’ is a dictionary variable that specifies the data types to use for each column. I figured out which data type takes up the least amount of memory for each column.

**Yachi**: Using less memory to store data allows that memory to be used to process data. It helps keep processing speeds faster when working with the large datasets.

**Hinata**: How’d you know which columns to keep and what data type to use?

**Kageyama**: It’s obvious when you look at the data.

**Hinata**: Kaaageeeeyaaaamaaaa…

**Kageyama**: Load the data and look at it yourself, scrub! You obviously need the practice.

**Yamaguchi**: Suga-senpai asked us to show how we got our solutions and explain our reasoning so we could learn from each other.

**Kageyama**: It’s faster my way.

**Tsukishima**: You failed every class where the teacher required you to show your work, didn’t you?

**Kageyama**: Shut up!

**Kageyama**: And stop messing up my notebook. If you guys are going to ask dumbass questions, do it somewhere else.

(See 01b\_entities\_kageyamaanswersdumbassquestions.ipynb for an explaination on how Kageyama came up with the code below.)

## Standardize column order and concatenate dataframes

bahamas\_entity\_raw = bahamas\_entity\_raw[keep\_cols]  
offshore\_entity\_raw = offshore\_entity\_raw[keep\_cols]  
panama\_entity\_raw = panama\_entity\_raw[keep\_cols]  
paradise\_entity\_raw = paradise\_entity\_raw[keep\_cols]  
  
entity\_df = pd.concat([bahamas\_entity\_raw, offshore\_entity\_raw, panama\_entity\_raw, paradise\_entity\_raw], ignore\_index=True)

## Standardize formatting

* Null entries imported as a string ‘nan’ instead of an actual null value.
* Capitalization is all over the place. Standardized as title case.

entity\_df.replace('nan', np.nan, inplace=True)  
entity\_df['name'] = entity\_df['name'].str.title()

## Consolidate entries in ‘status’ column

entity\_df['status'] = entity\_df['status'].str.lower()  
entity\_df.loc[entity\_df['status'].str.contains('liquidation', na=False), 'status'] = 'in liquidation'  
entity\_df.loc[entity\_df['status'].str.contains('liquidated', na=False), 'status'] = 'liquidated'  
entity\_df.loc[entity\_df['status'].str.contains('resigned', na=False), 'status'] = 'resigned agent'  
entity\_df.loc[entity\_df['status'].str.contains('sundry', na=False), 'status'] = 'sundry account'  
entity\_df.loc[entity\_df['status'].str.contains('dissolved', na=False), 'status'] = 'dissolved'  
entity\_df.loc[entity\_df['status'].str.contains('struck|defunct|deregistered', na=False), 'status'] = 'struck / defunct / deregistered'   
entity\_df['status'] = entity\_df['status'].str.title()

## Format date

entity\_df.loc[  
 (entity\_df['incorporation\_date'].str.contains('[A-Z]{3}', na=False)) & ((entity\_df['incorporation\_date'].str[-4:] > '2018') | (entity\_df['incorporation\_date'].str[-4:] < '1800')  
 ), 'incorporation\_date'] = np.nan  
entity\_df['formatted\_date'] = pd.to\_datetime(entity\_df.loc[entity\_df['incorporation\_date'].str.contains(',', na=False), 'incorporation\_date'], format='%b %d, %Y')  
entity\_df['formatted\_date'] = pd.to\_datetime(entity\_df.loc[entity\_df['incorporation\_date'].str.contains('[A-Z]{3}', na=False), 'incorporation\_date'], format='%d-%b-%Y')  
  
entity\_df.drop(columns=['incorporation\_date'], inplace=True)  
entity\_df.rename(columns={'formatted\_date':'incorporation\_date'}, inplace=True)

## Address duplicates and worthless entries

* ‘counry\_codes’ and ‘countries’ are strings that are actually lists. Not standardized as to which value comes first in the list, causing duplicates when counting values.
* ‘jurisdiction’ and ‘jurisdiction\_description’ included worthless values that should be null

entity\_df['country\_codes'] = entity\_df.loc[entity\_df['country\_codes'].notnull(), 'country\_codes'].str.split(';').apply(lambda x: sorted(x))  
entity\_df['countries'] = entity\_df.loc[entity\_df['countries'].notnull(), 'countries'].str.split(';').apply(lambda x: sorted(x))  
entity\_df['jurisdiction\_description'] = entity\_df['jurisdiction\_description'].str.title()  
entity\_df.loc[entity\_df['jurisdiction\_description'].str.contains('Undetermined|Recorded in leaked files as "fund"'), 'jurisdiction\_description'] = np.nan  
entity\_df['jurisdiction'] = entity\_df['jurisdiction']  
entity\_df.loc[entity\_df['jurisdiction'] == 'XXX', 'jurisdiction'] = np.nan

entity\_df.head(3)

| **node\_id** | **sourceID** | **name** | **country\_codes** | **countries** | **jurisdiction\_description** | **jurisdiction** | **service\_provider** | **status** | **incorporation\_date** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 20003127 | Bahamas Leaks | Dalma Corporation Limited | NaN | NaN | Bahamas | BAH | NaN | NaN | 1990-11-30 |
| 20010494 | Bahamas Leaks | Asia Construction Corporation Limited | NaN | NaN | Bahamas | BAH | NaN | NaN | 1992-08-14 |
| 20010495 | Bahamas Leaks | Euro Logistics Limited | NaN | NaN | Bahamas | BAH | NaN | NaN | 1992-08-14 |

**Kageyama**: At the team’s insistance, we’re also removing ‘Not identified’ and ‘XXX’ from the ‘countries’ and ‘country\_code’ columns.

**Tsukishima**: And made you put in this disclaimer so you wouldn’t ‘accidentally’ take credit for it, Your Majesty.

**Kageyama**: Don’t call me that!!

**Tsukishima**: Yes, Your Highness. Anything you say, Sire.

**Kageyama**: !!!!

def clean\_country\_and\_code(df):  
 df['country\_codes'].replace('XXX', np.nan, inplace=True)  
 df['country\_codes'].str.replace(';XXX|XXX;', '')  
 df['country\_codes'] = df.loc[df['country\_codes'].notnull(), 'country\_codes'].str.split(';').apply(lambda x: sorted(x))  
 df['countries'].replace('Not identified', np.nan, inplace=True)  
 df['countries'].str.replace(';Not identified|Not identified;', '')  
 df['countries'] = df.loc[df['countries'].notnull(), 'countries'].str.split(';').apply(lambda x: sorted(x))  
   
clean\_country\_and\_code(entity\_df)

entity\_df.to\_csv('../data/intermediate/entities.csv')