Home Work #4

Submit a Jupyter Notebook containing Python 3 (3.9 recommended) code using (only) Pandas that performs some preliminary processing of a dataset stored in a CSV file containing information about selected research articles (see the example provided: `research.csv`).

§1. The user should be able to execute the function `process_articles(filename)` to start the program. The input parameter `filename` (string) is the input CSV file to be processed.

§2. It should use the Pandas method `read_csv()` to read the CSV file into a *dataframe* named `df1` using

```
df1 = pd.read_csv(filename, delimiter = ',', header = 0,
                  encoding_errors = 'ignore')
```

§3. It must then use Pandas (and not plain Python) to perform the following processing (which should not be tailored to the example input file) steps.

1. Maintaining the order of the rows, sequentially assign numeric identifiers 0, 1, 2, ..., and store these identifiers in a new column `ID`.
2. Convert all numeric `Publication Year` values to integers, e.g., 2020.0 and `'2020'` should become `2020`. If non-numeric, then store `-9999`; replace missing values by `0`.

   You must execute a statement of the following form:

   ```
   df1['Publication Year'] =
                   df1['Publication Year'].apply(lambda x: ... )
   ```
3. If an `Authors` field is the string `no author`, then replace it by an empty string. Count the number of such fields.

   You must use the simple method of extracting `Authors` into a *series*, applying the changes to it, and then using its index values to update the *dataframe*. (See Appendix.)
4. Represent each missing entry in `Abstract` by an empty string. Count the number of such fields. You must use the method of selecting and updating a *dataframe* directly. (See Appendix.)
5. Count the number of empty DOIs

   If there is at least one, then

   (a) replace each by an empty string; and

   (b) create a *dataframe* `df_empty` containing those rows and another named `df2` containing the rest of the rows in `df1`. The new *dataframe*s `df_empty` and `df2` should have identical columns but no identical rows.
   Execute the following:

   ```
   assert len(df1)==len(df_empty_doi)+len(df2), 'DF PARTITION ERROR'
   ```

(c) If an entry with a DOI matches one without in `Document Title`, then attach that DOI to the latter entry in a new column `'Possible DOI'`.
Count the number of such DOIs matched. Hint: Use `pd.merge()`, `pd.concatenate()`. See https://pandas.pydata.org/docs/user_guide/merging.html

6. Print a message of the form

```
Number missing: Authors = ...; Abstracts = ...;
                DOIs = ...; Number of DOIs matched = ...
```

§4. The *dataframes* `df1`, `df2`, and `df_empty` should be globally accessible.

Appendix: Examples

Extract column selection as a *series*, apply function, and restore in *dataframe*

```
df = pd.DataFrame({'C1': [10, 11, 11, 12, 13],
                   'C2': [50, 0, 0, 25, 0]})
s = df[df['C1'] %2 == 0]['C2']

s = s+3
df.loc[s.index, 'C2'] = s
df

s [:] = 3 # leads to a warning: slice being assigned... So, ...
s = df[df['C1'] %2 == 0]['C2'].copy()
df.loc[s.index, 'C2'] = s
df
```

Combined selection and update on *dataframe*

```
df = pd.DataFrame({'C1': [10, 11, 11, 12, 13],
                   'C2': [50, 0, 0, 25, 0]})
df

df.loc[df['C1'] %2 == 0, ['C1','C2'] ]

df.loc[(df['C1'] %2 == 0) & (df['C2'] > 40), ['C1','C2'] ]

df.loc[(df['C1'] %2 == 0) & (df['C2'] > 40), 'C2']

df.loc[(df['C1'] %2 == 0) & (df['C2'] > 40), ['C1','C2'] ] = 100
df

df.loc[(df['C1'] %2 == 0) & (df['C2'] > 40), 'C1']  = 0
df

df.loc[(df['C1'] %2 == 0) & (df['C2'] > 40), 'C2']  = 500
df
```