Data Cleaning - ICE

Name: Cabot Steward

DATA 3300

Exercise

Import the *masonrybldg.xls* dataset into Notebooks, then complete the data cleaning activities noted below. Once complete, you'll have a single Excel document that includes the scrubbed data. We will be performing all data cleaning activities using the pandas and numpy libraries!

In [91]:
```
# import required libraries - pandas and numpy
import pandas as pd
import numpy as np
```

In [92]:
```
mason = pd.read_excel('masonrybldg.xlsx')
```

In [93]:
```
# produce a dataframe heading
# pd.set_option('display.max_rows', None)
mason.head()
```

Out[93]:

	Unnamed: 0	ObsID	Preliminary Risk Category	Neighborhood	Address	Year Built	No. Stories	Retrofit Level	Building Use	Estimated Number of Occupants	Confi
0	NaN	19	High Risk	Capitol Hill	925 E Pike St	1916	1	Substantial Alteration	Public Assembly	101+	
1	NaN	40	Medium Risk	Capitol Hill	1621 12th Ave	1917	1	Substantial Alteration	Commercial/Office	101+	Pho
2	NaN	265	Medium Risk	Capitol Hill	1510 Melrose Ave	1930	2	Substantial Alteration	Commercial/Residential	11-100	Goog
3	NaN	95	Medium Risk	Alki-Admiral	1321 Harbor Ave SW	1915	1	No visible retrofit	Commercial	11-100	Pho
4	NaN	49	Medium Risk	Alki/Admiral	2124 California Ave SW	1928	3	No visible retrofit	Residential	11-100	Goog

1. **Remove any leading and trailing spaces from all text columns. *Note: The trim feature does not remove additional spaces between two words.***

In [94]:
```
mason['Neighborhood'] = mason['Neighborhood'].str.strip()
mason['Address'] = mason['Address'].str.strip()
mason['Retrofit Level'] = mason['Retrofit Level'].str.strip()
mason['Building Use'] = mason['Building Use'].str.strip()
mason['Confirmation Source'] = mason['Confirmation Source'].str.strip()

mason.head()
```

Out[94]:

	Unnamed: 0	ObsID	Preliminary Risk Category	Neighborhood	Address	Year Built	No. Stories	Retrofit Level	Building Use	Estimated Number of Occupants	Confi
0	NaN	19	High Risk	Capitol Hill	925 E Pike St	1916	1	Substantial Alteration	Public Assembly	101+	
1	NaN	40	Medium Risk	Capitol Hill	1621 12th Ave	1917	1	Substantial Alteration	Commercial/Office	101+	Pho
2	NaN	265	Medium Risk	Capitol Hill	1510 Melrose Ave	1930	2	Substantial Alteration	Commercial/Residential	11-100	Goog
3	NaN	95	Medium Risk	Alki-Admiral	1321 Harbor Ave SW	1915	1	No visible retrofit	Commercial	11-100	Pho
4	NaN	49	Medium Risk	Alki/Admiral	2124 California Ave SW	1928	3	No visible retrofit	Residential	11-100	Goog

2. **Eliminate any records that have no Address or Retrofit Level data.**

In [95]:
```
mason.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 710 entries, 0 to 709
Data columns (total 11 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Unnamed: 0                      0 non-null      float64
 1   ObsID                           710 non-null    int64
 2   Preliminary Risk Category       710 non-null    object
 3   Neighborhood                    710 non-null    object
 4   Address                         707 non-null    object
 5   Year Built                      710 non-null    int64
 6   No. Stories                     710 non-null    int64
 7   Retrofit Level                  706 non-null    object
 8   Building Use                    710 non-null    object
 9   Estimated Number of Occupants   708 non-null    object
 10  Confirmation Source             707 non-null    object
dtypes: float64(1), int64(3), object(7)
memory usage: 61.1+ KB
```

In [96]:
```
mason_full = mason[mason['Address'].notna()]
mason_full = mason_full[mason_full['Retrofit Level'].notna()]
len(mason_full)
```

Out[96]: 703

3. **Ensure that the labels for Neighborhood and Retrofit Level are consistent (i.e., there's shouldn't be different spellings, abbreviations, or just multiple ways of saying the same thing).**

In [97]:
```
# examine entries for neighborhood using .value_counts()

mason_full["Retrofit Level"].value_counts()
mason_full["Neighborhood"].value_counts()
```

Out[97]:
```
Neighborhood
Capitol Hill                139
Duwamish/SODO                79
Cascade/Eastlake             71
Belltown                     68
Ballard                      66
Downtown                     57
First Hill                   45
Greenwood/Phinney Ridge      29
Columbia City                27
Central Area/Squire Park     24
Green Lake                   20
Georgetown                   17
Fremont                      13
Judkins Park                 13
Fauntleroy/Seaview           11
Beacon Hill                   6
Interbay                      5
Cap Hill                      3
Cedar Park/Meadowbrook        2
Alki-Admiral                  2
Broadview/Bitter Lake         2
Alki/Admiral                  2
Cascade/Eastlak               1
Highland Park                 1
Name: count, dtype: int64
```

In [98]:
```
#replace redundant neighborhood values
mason_full["Neighborhood"].replace({"Cascade/Eastlak": "Cascade/Eastlake",
                                    "Alki-Admiral": "Alki/Admiral",
                                    "Cap Hill": "Capitol Hill"},
                                    inplace=True)
mason_full["Neighborhood"].value_counts()
```

```
C:\Users\tucke\AppData\Local\Temp\ipykernel_3552\203858825.py:2: FutureWarning: A value is trying to be set on a copy
of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which
we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or d
f[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  mason_full["Neighborhood"].replace({"Cascade/Eastlak": "Cascade/Eastlake",
```

Out[98]:
```
Neighborhood
Capitol Hill               142
Duwamish/SODO               79
Cascade/Eastlake            72
Belltown                    68
Ballard                     66
Downtown                    57
First Hill                  45
Greenwood/Phinney Ridge     29
Columbia City               27
Central Area/Squire Park    24
Green Lake                  20
Georgetown                  17
Fremont                     13
Judkins Park                13
Fauntleroy/Seaview          11
Beacon Hill                  6
Interbay                     5
Alki/Admiral                 4
Broadview/Bitter Lake        2
Cedar Park/Meadowbrook       2
Highland Park                1
Name: count, dtype: int64
```

In [99]:
```
# examine entries of Retrofit Level using .value_counts()
mason_full["Retrofit Level"].value_counts()
```

Out[99]:
```
Retrofit Level
No visible retrofit       373
Permitted Retrofit        126
Substantial Alteration     89
Visible retrofit           70
None visible               45
Name: count, dtype: int64
```

In [100…
```
# mason_full = mason_full.replace() #replace redundant retrofit levels
#examine entries of Retrofit Level using .value_counts()
mason_full["Retrofit Level"].replace({"None visible": "No visible retrofit"}, inplace=True)
mason_full["Retrofit Level"].value_counts()
```

```
C:\Users\tucke\AppData\Local\Temp\ipykernel_3552\2354037165.py:3: FutureWarning: A value is trying to be set on a cop
y of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which
we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or d
f[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  mason_full["Retrofit Level"].replace({"None visible": "No visible retrofit"}, inplace=True)
```

Out[100…
```
Retrofit Level
No visible retrofit       418
Permitted Retrofit        126
Substantial Alteration     89
Visible retrofit           70
Name: count, dtype: int64
```

4. **Many of the buildings are dual-use. This is indicated in the Building Use column. Create two separate columns from the Building Use column, one for the first use listed and the other for the second.**

In [101…
```
mason_full['Building Use'].value_counts() # preview entries for Building Use
# switching to value_counts as it is more helpful to see
```

Out[101…

```
Building Use
Commercial                206
Residential               125
Commercial/Office          94
Commercial/Residential     88
Public Assembly            78
Office                     36
Schools                    30
Industrial                 17
Other Mixed Uses           17
Government                 10
Vacant                      1
Emergency                   1
Name: count, dtype: int64
```

In [102…

```
mason_full[['Primary Use', 'Secondary Use']] = mason_full['Building Use'].str.split('/', expand = True) # split build
mason_full.head()
```

Out[102…

	Unnamed: 0	ObsID	Preliminary Risk Category	Neighborhood	Address	Year Built	No. Stories	Retrofit Level	Building Use	Estimated Number of Occupants	Confi
0	NaN	19	High Risk	Capitol Hill	925 E Pike St	1916	1	Substantial Alteration	Public Assembly	101+	
1	NaN	40	Medium Risk	Capitol Hill	1621 12th Ave	1917	1	Substantial Alteration	Commercial/Office	101+	Pho
2	NaN	265	Medium Risk	Capitol Hill	1510 Melrose Ave	1930	2	Substantial Alteration	Commercial/Residential	11-100	Goog
3	NaN	95	Medium Risk	Alki/Admiral	1321 Harbor Ave SW	1915	1	No visible retrofit	Commercial	11-100	Pho
4	NaN	49	Medium Risk	Alki/Admiral	2124 California Ave SW	1928	3	No visible retrofit	Residential	11-100	Goog

5. **Create a new column called "IsCritical". For those buildings shown with a preliminary risk value of "Critical Risk", the value for "IsCritical" should be 1. For all others, the value should be 0.**

In [103…

```
#view categories of Preliminary Risk Category using .value_counts()
mason_full['Preliminary Risk Category'].value_counts()
```

Out[103…

```
Preliminary Risk Category
Medium Risk      563
High Risk        109
Critical Risk     31
Name: count, dtype: int64
```

In [104…
```
mason_full["IsCritical"] = np.where(mason_full['Preliminary Risk Category'] == 'Critical Risk', 1, 0)
mason_full.head()
```

Out[104…

	Unnamed: 0	ObsID	Preliminary Risk Category	Neighborhood	Address	Year Built	No. Stories	Retrofit Level	Building Use	Estimated Number of Occupants	Confi
0	NaN	19	High Risk	Capitol Hill	925 E Pike St	1916	1	Substantial Alteration	Public Assembly	101+	
1	NaN	40	Medium Risk	Capitol Hill	1621 12th Ave	1917	1	Substantial Alteration	Commercial/Office	101+	Pho
2	NaN	265	Medium Risk	Capitol Hill	1510 Melrose Ave	1930	2	Substantial Alteration	Commercial/Residential	11-100	Goog
3	NaN	95	Medium Risk	Alki/Admiral	1321 Harbor Ave SW	1915	1	No visible retrofit	Commercial	11-100	Pho
4	NaN	49	Medium Risk	Alki/Admiral	2124 California Ave SW	1928	3	No visible retrofit	Residential	11-100	Goog

6. **We'd like to be able to categorize the buildings' age. Create a new column and name it Era. Populate this column with information reflecting to which of the following 'eras' each building belongs: "before 1920", "1920-1939", "1940-1959", "1960-1979", or "after 1979".**

In [109…
```
conditions = [
    (mason_full['Year Built'] < 1920),
    (mason_full['Year Built'] <= 1939),
    (mason_full['Year Built'] <= 1959),
    (mason_full['Year Built'] <= 1979),
```

```
    (mason_full['Year Built'] > 1979)
]
# you must add <= or you get incorrect values

values = ['before 1920', '1920-1939', '1940-1959', '1960-1979', 'after 1979']
mason_full['Era'] = np.select(conditions, values) # create new era variable
mason_full.head()
```

Out[109…

	Unnamed: 0	ObsID	Preliminary Risk Category	Neighborhood	Address	Year Built	No. Stories	Retrofit Level	Building Use	Estimated Number of Occupants	Confi
0	NaN	19	High Risk	Capitol Hill	925 E Pike St	1916	1	Substantial Alteration	Public Assembly	101+	
1	NaN	40	Medium Risk	Capitol Hill	1621 12th Ave	1917	1	Substantial Alteration	Commercial/Office	101+	Pho
2	NaN	265	Medium Risk	Capitol Hill	1510 Melrose Ave	1930	2	Substantial Alteration	Commercial/Residential	11-100	Goog
3	NaN	95	Medium Risk	Alki/Admiral	1321 Harbor Ave SW	1915	1	No visible retrofit	Commercial	11-100	Pho
4	NaN	49	Medium Risk	Alki/Admiral	2124 California Ave SW	1928	3	No visible retrofit	Residential	11-100	Goog

7. **Delete any unnecessary columns.**

In [110…

```
mason_full = mason_full.drop(columns=["Unnamed: 0", "Building Use"])
```

8. **Sort the data.**

In [111…
```
mason_full = mason_full.set_index('ObsID') # set index to ObsID
mason_full = mason_full.sort_index() # sort by index
mason_full.head()
```

Out[111…

ObsID	Preliminary Risk Category	Neighborhood	Address	Year Built	No. Stories	Retrofit Level	Estimated Number of Occupants	Confirmation Source	Primary Use	Secondary Use	IsCr
1	Medium Risk	Cascade/Eastlake	305 Bell St	1925	1	No visible retrofit	11-100	Google Street View	Commercial	None	
2	Medium Risk	Ballard	2016 NW Market St	1906	2	No visible retrofit	11-100	Field Visit	Commercial	Office	
3	Medium Risk	Capitol Hill	225 Broadway E	1940	2	No visible retrofit	11-100	Field Visit	Commercial	None	
4	Medium Risk	Cascade/Eastlake	2132 5th Ave	1922	4	No visible retrofit	11-100	Google Street View	Residential	None	
5	Medium Risk	Cascade/Eastlake	1814 Minor Ave	1905	3	Permitted Retrofit	11-100	Permit Drawing Review	Residential	None	

9. **Export the cleaned data as an Excel file. Save this .ipynb file, print it to a PDF, and download the Excel file; you'll upload all three files to this Canvas assignment!**

In [112…
```
mason_full.to_excel('cleaned dataset.xlsx') #export cleaned data to excel
```