In [1]:

```
#Q1a
import pandas as pd
df1 = pd.read_csv("data1.csv", index_col=0)
print(df1)
df2 = pd.read_csv("data2.csv", index_col=1)
print(df2)
```

```
   A    B  C
0  1  2.0  3
1  4  5.0  6
4  2  NaN  5
       A  B  C
2      2  5  6
3  Hello  3  4
```

In [2]:

```
#Q1b
df3 = pd.concat([df1, df2])
print(df3)
```

```
       A    B  C
0      1  2.0  3
1      4  5.0  6
4      2  NaN  5
2      2  5.0  6
3  Hello  3.0  4
```

In [3]:

```
#Q1c
df4 = pd.read_csv("data3.csv", index_col=0)
print(df4)
df5 = pd.concat([df3, df4], axis=1)
print(df5)
```

```
   D  E
1  1  7
4  0  8
       A    B  C    D    E
0      1  2.0  3  NaN  NaN
1      4  5.0  6  1.0  7.0
2      2  5.0  6  NaN  NaN
3  Hello  3.0  4  NaN  NaN
4      2  NaN  5  0.0  8.0
```

In [4]:

```
#Q1d
df6 = pd.read_json("data.json")
# print(df6)
df7 = pd.concat([df5, df6], axis=0)
print(df7)
```

```
       A    B    C    D    E
0      1  2.0  3.0  NaN  NaN
1      4  5.0  6.0  1.0  7.0
2      2  5.0  6.0  NaN  NaN
3  Hello  3.0  4.0  NaN  NaN
4      2  NaN  5.0  0.0  8.0
0     11  9.0  NaN  NaN  NaN
1     22  7.0  NaN  NaN  NaN
2     33  8.0  NaN  NaN  NaN
```

In [5]:

```
#Q1e
```

```
import numpy as np
# df7 = df7.replace('Hello', np.nan)
# df7 = df7.replace(r'^[a-zA-Z]', np.nan, regex=True)
# df7
df7 = df7.apply(pd.to_numeric,errors="coerce")
df7
```

Out[5]:

	A	B	C	D	E
0	1.0	2.0	3.0	NaN	NaN
1	4.0	5.0	6.0	1.0	7.0
2	2.0	5.0	6.0	NaN	NaN
3	NaN	3.0	4.0	NaN	NaN
4	2.0	NaN	5.0	0.0	8.0
0	11.0	9.0	NaN	NaN	NaN
1	22.0	7.0	NaN	NaN	NaN
2	33.0	8.0	NaN	NaN	NaN

In [6]:

```
#Q1f
df7 = df7.fillna(df7.mean())
df7
```

Out[6]:

	A	B	C	D	E
0	1.000000	2.000000	3.0	0.5	7.5
1	4.000000	5.000000	6.0	1.0	7.0
2	2.000000	5.000000	6.0	0.5	7.5
3	10.714286	3.000000	4.0	0.5	7.5
4	2.000000	5.571429	5.0	0.0	8.0
0	11.000000	9.000000	4.8	0.5	7.5
1	22.000000	7.000000	4.8	0.5	7.5
2	33.000000	8.000000	4.8	0.5	7.5

Dropping Columns in a DataFrame

In [7]:

```
#Q2
df = pd.read_csv('BL-Flickr-Images-Book.csv')
df.head()
```

Out[7]:

	Identifier	Edition Statement	Place of Publication	Date of Publication	Publisher	Title	Author	Contributors	Corporate Author	Corporate Contributors	Fo ov
0	206	NaN	London	1879 [1878]	S. Tinsley & Co.	Walter Forbes. [A novel.] By A. A	A. A.	FORBES, Walter.	NaN	NaN	
1	216	NaN	London; Virtue & Yorston	1868	Virtue & Co.	All for Greed. [A novel. The dedication signed...	A., A. A.	BLAZE DE BURY, Marie Pauline Rose -	NaN	NaN	

	Identifier	Edition Statement	Place of Publication	Date of Publication	Publisher	Title	Author	Baroness Contributors BLAZE DE	Corporate Author	Corporate Contributors	Fo ov
2	218	NaN	London	1869	Bradbury, Evans & Co.	Love the Avenger. By the author of “All for Gr...	A., A. A.	BURY, Marie Pauline Rose - Baroness	NaN	NaN	
3	472	NaN	London	1851	James Darling	Welsh Sketches, chiefly ecclesiastical, to the...	A., E. S.	Appleyard, Ernest Silvanus.	NaN	NaN	
4	480	A new edition, revised, etc.	London	1857	Wertheim & Macintosh	[The World in which I live, and my place in it...	A., E. S.	BROOME, John Henry.	NaN	NaN	

In [8]:

```
to_drop = ['Edition Statement',
           'Corporate Author',
           'Corporate Contributors',
           'Former owner',
           'Engraver',
           'Contributors',
           'Issuance type',
           'Shelfmarks']

df = df.drop(to_drop, axis=1)
```

In [9]:

```
df.head()
```

Out[9]:

	Identifier	Place of Publication	Date of Publication	Publisher	Title	Author	Flickr URL
0	206	London	1879 [1878]	S. Tinsley & Co.	Walter Forbes. [A novel.] By A. A	A. A.	http://www.flickr.com/photos/britishlibrary/ta...
1	216	London; Virtue & Yorston	1868	Virtue & Co.	All for Greed. [A novel. The dedication signed...	A., A. A.	http://www.flickr.com/photos/britishlibrary/ta...
2	218	London	1869	Bradbury, Evans & Co.	Love the Avenger. By the author of “All for Gr...	A., A. A.	http://www.flickr.com/photos/britishlibrary/ta...
3	472	London	1851	James Darling	Welsh Sketches, chiefly ecclesiastical, to the...	A., E. S.	http://www.flickr.com/photos/britishlibrary/ta...
4	480	London	1857	Wertheim & Macintosh	[The World in which I live, and my place in it...	A., E. S.	http://www.flickr.com/photos/britishlibrary/ta...

Changing the Index of a DataFrame

In [10]:

```
df['Identifier'].is_unique #to check if all identifies are uniques, can be used to check uniqueness if we want to make it index
```

Out[10]:

```
True
```

In [11]:

```
df = df.set_index('Identifier')
df.head()
```

Out[11]:

Identifier	Place of Publication	Date of Publication	Publisher	Title	Author	Flickr URL
206	London	1879 [1878]	S. Tinsley & Co.	Walter Forbes. [A novel.] By A. A	A. A.	http://www.flickr.com/photos/britishlibrary/ta...
216	London; Virtue & Yorston	1868	Virtue & Co.	All for Greed. [A novel. The dedication signed...	A., A. A.	http://www.flickr.com/photos/britishlibrary/ta...
218	London	1869	Bradbury, Evans & Co.	Love the Avenger. By the author of “All for Gr...	A., A. A.	http://www.flickr.com/photos/britishlibrary/ta...
472	London	1851	James Darling	Welsh Sketches, chiefly ecclesiastical, to the...	A., E. S.	http://www.flickr.com/photos/britishlibrary/ta...
480	London	1857	Wertheim & Macintosh	[The World in which I live, and my place in it...	A., E. S.	http://www.flickr.com/photos/britishlibrary/ta...

In [12]:

```
df.loc[206]
```

Out[12]:

```
Place of Publication                                          London
Date of Publication                                     1879 [1878]
Publisher                                          S. Tinsley & Co.
Title                                 Walter Forbes. [A novel.] By A. A
Author                                                        A. A.
Flickr URL              http://www.flickr.com/photos/britishlibrary/ta...
Name: 206, dtype: object
```

Tidying up Fields in the Data

In [13]:

```
df.dtypes.value_counts() # df.get_dtypes_counts() is deprecated
```

Out[13]:

```
object    6
dtype: int64
```

In [14]:

```
df.loc[1905:, 'Date of Publication'].head(10)
```

Out[14]:

```
Identifier
1905           1888
1929    1839, 38-54
2836           1897
2854           1865
2956        1860-63
2957           1873
3017           1866
3131           1899
```

```
4598          1814
4884          1820
Name: Date of Publication, dtype: object
```

In [15]:

```
extr = df['Date of Publication'].str.extract(r'^(\d{4})', expand=False)
extr.head()
```

Out[15]:

```
Identifier
206    1879
216    1868
218    1869
472    1851
480    1857
Name: Date of Publication, dtype: object
```

In [16]:

```
df['Date of Publication'] = pd.to_numeric(extr)
df['Date of Publication'].dtype
```

Out[16]:

```
dtype('float64')
```

In [17]:

```
df['Date of Publication'].isnull().sum() / len(df) #to check how many values are null in the column
```

Out[17]:

```
0.11717147339205986
```

In [18]:

```
df.loc[4157862]
```

Out[18]:

```
Place of Publication                                  Newcastle-upon-Tyne
Date of Publication                                                  1867
Publisher                                                     T. Fordyce
Title                  Local Records; or, Historical Register of rema...
Author                     FORDYCE, T. - Printer, of Newcastle-upon-Tyne
Flickr URL             http://www.flickr.com/photos/britishlibrary/ta...
Name: 4157862, dtype: object
```

In [19]:

```
df.loc[4159587]
```

Out[19]:

```
Place of Publication                                  Newcastle upon Tyne
Date of Publication                                                  1834
Publisher                                                Mackenzie & Dent
Title                  An historical, topographical and descriptive v...
Author                                              Mackenzie, E. (Eneas)
Flickr URL             http://www.flickr.com/photos/britishlibrary/ta...
Name: 4159587, dtype: object
```

In [20]:

```
pub = df['Place of Publication']
london = pub.str.contains('London') #will make rows with string 'London' True
london[:5] #print from 0 till 5th index
```

Out[20]:

```
Identifier
```

```
206     True
216     True
218     True
472     True
480     True
Name: Place of Publication, dtype: bool
```

In [21]:

```
oxford = pub.str.contains('Oxford')
oxford[:10]
```

Out[21]:

```
Identifier
206     False
216     False
218     False
472     False
480     False
481     False
519     False
667      True
874     False
1143    False
Name: Place of Publication, dtype: bool
```

Combining str Methods with NumPy to Clean Columns

In [22]:

```
df['Place of Publication'].head(10)
```

Out[22]:

```
Identifier
206                                    London
216                 London; Virtue & Yorston
218                                    London
472                                    London
480                                    London
481                                    London
519                                    London
667     pp. 40. G. Bryan & Co: Oxford, 1898
874                                   London]
1143                                   London
Name: Place of Publication, dtype: object
```

In [23]:

```
df['Place of Publication'] = np.where(london, 'London',
                                      np.where(oxford, 'Oxford', pub.str.replace('-',' '
)))
df['Place of Publication'].head(10)
```

Out[23]:

```
Identifier
206     London
216     London
218     London
472     London
480     London
481     London
519     London
667     Oxford
874     London
1143    London
Name: Place of Publication, dtype: object
```

Cleaning the Entire Dataset Using the applymap Function

In [24]:

```
university_towns = []
with open('university_towns.txt') as file:
    for line in file:
        if '[edit]' in line:
            #remember this state until next is found
            state = line
        else:
            # Otherwise, we have a city; keep state as last-seen
            university_towns.append((state, line))

university_towns[:5]
```

Out[24]:

```
[('Alabama[edit]\n', 'Auburn (Auburn University)[1]\n'),
 ('Alabama[edit]\n', 'Florence (University of North Alabama)\n'),
 ('Alabama[edit]\n', 'Jacksonville (Jacksonville State University)[2]\n'),
 ('Alabama[edit]\n', 'Livingston (University of West Alabama)[2]\n'),
 ('Alabama[edit]\n', 'Montevallo (University of Montevallo)[2]\n')]
```

In [25]:

```
towns_df = pd.DataFrame(university_towns, columns=["State", "Region_Name"])
```

In [26]:

```
towns_df.head(10)
```

Out[26]:

	State	Region_Name
0	Alabama[edit]\n	Auburn (Auburn University)[1]\n
1	Alabama[edit]\n	Florence (University of North Alabama)\n
2	Alabama[edit]\n	Jacksonville (Jacksonville State University)[2]\n
3	Alabama[edit]\n	Livingston (University of West Alabama)[2]\n
4	Alabama[edit]\n	Montevallo (University of Montevallo)[2]\n
5	Alabama[edit]\n	Troy (Troy University)[2]\n
6	Alabama[edit]\n	Tuscaloosa (University of Alabama, Stillman Co...
7	Alabama[edit]\n	Tuskegee (Tuskegee University)[5]\n
8	Alaska[edit]\n	Fairbanks (University of Alaska Fairbanks)[2]\n
9	Arizona[edit]\n	Flagstaff (Northern Arizona University)[6]\n

In [27]:

```
def get_citystate(item):
    if ' (' in item:
        return item[:item.find(' (')]
    elif '[' in item:
        return item[:item.find('[')]
    else:
        return item
```

In [28]:

```
towns_df =  towns_df.applymap(get_citystate)
```

In [29]:

```
towns_df.head(10)
```

Out[29]:

Out[29]:

	State	Region_Name
0	Alabama	Auburn
1	Alabama	Florence
2	Alabama	Jacksonville
3	Alabama	Livingston
4	Alabama	Montevallo
5	Alabama	Troy
6	Alabama	Tuscaloosa
7	Alabama	Tuskegee
8	Alaska	Fairbanks
9	Arizona	Flagstaff

Renaming Columns and Skipping Rows

In [30]:

```
olympics_df = pd.read_csv('olympics.csv')
olympics_df.head()
```

Out[30]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	NaN	? Summer	01 !	02 !	03 !	Total	? Winter	01 !	02 !	03 !	Total	? Games	01 !	02 !	03 !	Combined total
1	Afghanistan (AFG)	13	0	0	2	2	0	0	0	0	0	13	0	0	2	2
2	Algeria (ALG)	12	5	2	8	15	3	0	0	0	0	15	5	2	8	15
3	Argentina (ARG)	23	18	24	28	70	18	0	0	0	0	41	18	24	28	70
4	Armenia (ARM)	5	1	2	9	12	6	0	0	0	0	11	1	2	9	12

In [31]:

```
olympics_df.loc[0]
#rough work
#search for apply() vs applymap()
#search for iloc[] vs loc[]
```

Out[31]:

```
0                 NaN
1           ? Summer
2               01 !
3               02 !
4               03 !
5              Total
6           ? Winter
7               01 !
8               02 !
9               03 !
10             Total
11          ? Games
12              01 !
13              02 !
14              03 !
15    Combined total
Name: 0, dtype: object
```

In [32]:

```
olympics_df = pd.read_csv('olympics.csv', header=1)
olympics_df.head()
```

Out[32]:

	Unnamed: 0	? Summer	01 !	02 !	03 !	Total	? Winter	01 !.1	02 !.1	03 !.1	Total.1	? Games	01 !.2	02 !.2	03 !.2	Combined total
0	Afghanistan (AFG)	13	0	0	2	2	0	0	0	0	0	13	0	0	2	2
1	Algeria (ALG)	12	5	2	8	15	3	0	0	0	0	15	5	2	8	15
2	Argentina (ARG)	23	18	24	28	70	18	0	0	0	0	41	18	24	28	70
3	Armenia (ARM)	5	1	2	9	12	6	0	0	0	0	11	1	2	9	12
4	Australasia (ANZ) [ANZ]	2	3	4	5	12	0	0	0	0	0	2	3	4	5	12

In [33]:

```
new_names =    {'Unnamed: 0': 'Country',
                '? Summer': 'Summer Olympics',
                '01 !': 'Gold',
                '02 !': 'Silver',
                '03 !': 'Bronze',
                '? Winter': 'Winter Olympics',
                '01 !.1': 'Gold.1',
                '02 !.1': 'Silver.1',
                '03 !.1': 'Bronze.1',
                '? Games': '# Games',
                '01 !.2': 'Gold.2',
                '02 !.2': 'Silver.2',
                '03 !.2': 'Bronze.2'}
olympics_df = olympics_df.rename(columns=new_names)
```

In [34]:

```
olympics_df.head()
```

Out[34]:

	Country	Summer Olympics	Gold	Silver	Bronze	Total	Winter Olympics	Gold.1	Silver.1	Bronze.1	Total.1	# Games	Gold.2	Silver.2
0	Afghanistan (AFG)	13	0	0	2	2	0	0	0	0	0	13	0	0
1	Algeria (ALG)	12	5	2	8	15	3	0	0	0	0	15	5	2
2	Argentina (ARG)	23	18	24	28	70	18	0	0	0	0	41	18	24
3	Armenia (ARM)	5	1	2	9	12	6	0	0	0	0	11	1	2
4	Australasia (ANZ) [ANZ]	2	3	4	5	12	0	0	0	0	0	2	3	4

In [ ]: