In this section of jupyter notebook, I will work on cost_of_living_df. Lets start the data wrangling with this dataframe.

In [2]:
```
import pandas as pd
```

In [4]:
```
cost_of_living_df = pd.read_csv("/Users/ertuboston/Documents/Data_Science_Merrimack/DSE5002/P
```

In [6]:
```
cost_of_living_df.head(6)
```

Out[6]:

	Rank	City	Cost of Living Index	Rent Index	Cost of Living Plus Rent Index	Groceries Index	Restaurant Price Index	Local Purchasing Power Index
0	NaN	Hamilton, Bermuda	149.02	96.10	124.22	157.89	155.22	79.43
1	NaN	Zurich, Switzerland	131.24	69.26	102.19	136.14	132.52	129.79
2	NaN	Basel, Switzerland	130.93	49.38	92.70	137.07	130.95	111.53
3	NaN	Zug, Switzerland	128.13	72.12	101.87	132.61	130.93	143.40
4	NaN	Lugano, Switzerland	123.99	44.99	86.96	129.17	119.80	111.96
5	NaN	Lausanne, Switzerland	122.03	59.55	92.74	122.56	127.01	127.01

In [8]:
```
### First lets separate the City into two column as city and country
### then drop the City column

split_result = cost_of_living_df['City'].str.split(',', expand = True)
cost_of_living_df[['city','country']] = split_result.iloc[:, :2]
```

In [10]:
```
cost_of_living_df.drop(columns =['City'], inplace = True)
```

In [11]:
```
### Lets check it out
print(cost_of_living_df.columns)
```

```
Index(['Rank', 'Cost of Living Index', 'Rent Index',
       'Cost of Living Plus Rent Index', 'Groceries Index',
       'Restaurant Price Index', 'Local Purchasing Power Index', 'city',
       'country'],
      dtype='object')
```

In [12]:
```
### In any case, if there are any leading or trailing space in the city or country column,
### we can use str.strip() function to clean it.
### We will apply it to city and country columns

cost_of_living_df['city'] = cost_of_living_df['city'].str.strip()
cost_of_living_df['country'] = cost_of_living_df['country'].str.strip()
```

In [13]:
```
### Lets check the city column as unique list.abs

city_unique_values = cost_of_living_df['city'].unique()
city_unique_values.sort()
# city_unique_values
### If you run the last code, you see that city column is good, no need to clean it.
### There are no New York, new york types of duplicates.
```

In [18]:

```
### Lets do the same thing for country column

country_unique_values = cost_of_living_df['country'].unique()
country_unique_values.sort()
country_unique_values

### I see that in the country section, we have state abbreviations mixed with country names
### We can replace those state abbreviations with the United States of America.
```

Out[18]:
```
array(['AK', 'AL', 'AR', 'AZ', 'Afghanistan', 'Albania', 'Algeria',
       'Argentina', 'Armenia', 'Australia', 'Austria', 'Azerbaijan', 'BC',
       'Bahamas', 'Bahrain', 'Bangladesh', 'Belarus', 'Belgium',
       'Bermuda', 'Bolivia', 'Bosnia And Herzegovina', 'Botswana',
       'Brazil', 'Bulgaria', 'CA', 'CO', 'Cambodia', 'Canada', 'Chile',
       'China', 'Colombia', 'Costa Rica', 'Croatia', 'Cuba', 'Cyprus',
       'Czech Republic', 'DC', 'Denmark', 'Dominican Republic', 'Ecuador',
       'Egypt', 'El Salvador', 'Estonia', 'Ethiopia', 'FL', 'Fiji',
       'Finland', 'France', 'GA', 'Georgia', 'Germany', 'Ghana', 'Greece',
       'Guatemala', 'HI', 'Hong Kong', 'Hungary', 'IA', 'ID', 'IL', 'IN',
       'Iceland', 'India', 'Indonesia', 'Iran', 'Iraq', 'Ireland',
       'Israel', 'Italy', 'Ivory Coast', 'Jamaica', 'Japan', 'Jersey',
       'Jordan', 'KS', 'KY', 'Kazakhstan', 'Kenya',
       'Kosovo (Disputed Territory)', 'Kuwait', 'Kyrgyzstan', 'LA',
       'Latvia', 'Lebanon', 'Lithuania', 'Luxembourg', 'MA', 'MD', 'MI',
       'MN', 'MO', 'Macao', 'Malaysia', 'Maldives', 'Malta', 'Mexico',
       'Moldova', 'Mongolia', 'Montenegro', 'Morocco', 'NC', 'NE', 'NJ',
       'NM', 'NV', 'NY', 'Nepal', 'Netherlands', 'New Zealand',
       'Newfoundland and Labrador', 'Nigeria', 'North Macedonia',
       'Norway', 'OH', 'OK', 'OR', 'Oman', 'PA', 'Pakistan', 'Panama',
       'Paraguay', 'Peru', 'Philippines', 'Poland', 'Portugal',
       'Puerto Rico', 'Qatar', 'Romania', 'Russia', 'Rwanda', 'SC',
       'Saudi Arabia', 'Selangor', 'Senegal', 'Serbia', 'Shandong',
       'Singapore', 'Slovakia', 'Slovenia', 'South Africa', 'South Korea',
       'Spain', 'Sri Lanka', 'Suriname', 'Sweden', 'Switzerland', 'Syria',
       'TN', 'TX', 'Taiwan', 'Tanzania', 'Thailand',
       'Trinidad And Tobago', 'Tunisia', 'Turkey', 'UT', 'Uganda',
       'Ukraine', 'United Arab Emirates', 'United Kingdom', 'Uruguay',
       'Uzbekistan', 'VA', 'Venezuela', 'Vietnam', 'WA', 'WI', 'Zambia',
       'Zimbabwe'], dtype=object)
```

In [19]:
```
### Replacing State abbreviations with United States of America.
```

```
cost_of_living_df['country'] = cost_of_living_df['country'].replace(["AK", "AL", "AR", "AZ",
    "ID", "IL", "IN", "KS", "KY", "LA", "MA", "MD", "ME", "MI", "MN", "MO",
    "MS", "MT", "NC", "ND", "NE", "NH", "NJ", "NM", "NV", "NY", "OH", "OK",
    "OR", "PA", "RI", "SC", "SD", "TN", "TX", "UT", "VA", "VT", "WA", "WI",
    "DC", "AS", "GU", "MP", "PR", "VI", "WV", "WY"], "United States of America")
```

In [22]:

```
### Lets check it out
cost_of_living_df['country'].unique()
```

Out[22]:
```
array(['Bermuda', 'Switzerland', 'Lebanon', 'Norway',
       'United States of America', 'Iceland', 'Jersey', 'Israel',
       'Denmark', 'Bahamas', 'United Kingdom', 'Japan', 'France',
       'Singapore', 'Australia', 'Luxembourg', 'Finland', 'Netherlands',
       'Hong Kong', 'BC', 'New Zealand', 'Ireland', 'Sweden',
       'South Korea', 'Germany', 'Newfoundland and Labrador', 'Austria',
       'Canada', 'Belgium', 'Italy', 'Malta', 'Puerto Rico', 'Macao',
       'Taiwan', 'Cyprus', 'United Arab Emirates', 'Spain', 'Qatar',
       'Trinidad And Tobago', 'Greece', 'Maldives', 'Slovenia', 'Cuba',
       'Estonia', 'Panama', 'Bahrain', 'China', 'Saudi Arabia',
       'Philippines', 'Jordan', 'Uruguay', 'Czech Republic', 'Portugal',
       'Croatia', 'Jamaica', 'Latvia', 'Oman', 'Senegal', 'Ethiopia',
       'Thailand', 'Cambodia', 'Slovakia', 'Suriname', 'Kuwait',
       'Costa Rica', 'Ivory Coast', 'Lithuania', 'Hungary', 'Zimbabwe',
       'Chile', 'El Salvador', 'Venezuela', 'South Africa',
       'Dominican Republic', 'Guatemala', 'Poland', 'Indonesia',
       'Botswana', 'Bulgaria', 'Russia', 'Ecuador', 'Romania', 'Serbia',
       'Malaysia', 'Morocco', 'Montenegro', 'Fiji', 'Vietnam', 'Mexico',
       'Ghana', 'Albania', 'Bosnia And Herzegovina', 'Iraq', 'Bolivia',
       'Iran', 'Brazil', 'Nigeria', 'Syria', 'Uganda', 'Kenya',
       'Shandong', 'Argentina', 'Bangladesh', 'North Macedonia',
       'Mongolia', 'Peru', 'Ukraine', 'India', 'Armenia', 'Tanzania',
       'Sri Lanka', 'Zambia', 'Belarus', 'Egypt', 'Rwanda', 'Moldova',
       'Azerbaijan', 'Turkey', 'Georgia', 'Paraguay', 'Kazakhstan',
       'Tunisia', 'Nepal', 'Algeria', 'Uzbekistan', 'Colombia',
       'Kyrgyzstan', 'Kosovo (Disputed Territory)', 'Selangor',
       'Pakistan', 'Afghanistan'], dtype=object)
```

In [23]:
```
### Now, let's check for any NA values in columns.
cost_of_living_df.isna().sum()

### There are NAs in the Rank column which won't affect our research.
### We can drop that column off.
```

Out[23]:
```
Rank                                578
Cost of Living Index                  0
Rent Index                            0
Cost of Living Plus Rent Index        0
Groceries Index                       0
Restaurant Price Index                0
Local Purchasing Power Index          0
city                                  0
country                               0
dtype: int64
```

In [26]:
```
cost_of_living_df.drop(columns = ['Rank'], inplace = True)
```

In [28]:
```
cost_of_living_df
```

Out[28]:

	Cost of Living Index	Rent Index	Cost of Living Plus Rent Index	Groceries Index	Restaurant Price Index	Local Purchasing Power Index	city	country
0	149.02	96.10	124.22	157.89	155.22	79.43	Hamilton	Bermuda
1	131.24	69.26	102.19	136.14	132.52	129.79	Zurich	Switzerland
2	130.93	49.38	92.70	137.07	130.95	111.53	Basel	Switzerland
3	128.13	72.12	101.87	132.61	130.93	143.40	Zug	Switzerland
4	123.99	44.99	86.96	129.17	119.80	111.96	Lugano	Switzerland
...	...	...	...	...	...	...	...	...
573	20.79	3.60	12.73	22.19	13.31	38.83	Kanpur	India
574	20.75	4.84	13.29	18.48	15.21	29.16	Karachi	Pakistan
575	20.52	4.78	13.14	18.51	16.18	22.91	Rawalpindi	Pakistan
576	18.68	2.94	11.30	18.37	11.80	25.09	Multan	Pakistan
577	18.55	2.37	10.97	16.62	14.39	26.00	Peshawar	Pakistan

578 rows × 8 columns

In [30]:
```
cost_of_living_df.head(5)
```

Out[30]:

	Cost of Living Index	Rent Index	Cost of Living Plus Rent Index	Groceries Index	Restaurant Price Index	Local Purchasing Power Index	city	country
0	149.02	96.10	124.22	157.89	155.22	79.43	Hamilton	Bermuda
1	131.24	69.26	102.19	136.14	132.52	129.79	Zurich	Switzerland
2	130.93	49.38	92.70	137.07	130.95	111.53	Basel	Switzerland
3	128.13	72.12	101.87	132.61	130.93	143.40	Zug	Switzerland
4	123.99	44.99	86.96	129.17	119.80	111.96	Lugano	Switzerland

In [32]:

```
### I would like to move the city and country column to the first 0 and 1 index.
### For this we will first pop the two columns and insert it in the place we want

### Step 1: Store the column data and remove it from its original position
city_col = cost_of_living_df.pop('city')
country_col = cost_of_living_df.pop('country')
### Step 2: Insert the column back into data frame at the new position

cost_of_living_df.insert(0,'City', city_col)
cost_of_living_df.insert(1,'Country',country_col)

cost_of_living_df.head(3)
```

Out[32]:

	City	Country	Cost of Living Index	Rent Index	Cost of Living Plus Rent Index	Groceries Index	Restaurant Price Index	Local Purchasing Power Index
0	Hamilton	Bermuda	149.02	96.10	124.22	157.89	155.22	79.43
1	Zurich	Switzerland	131.24	69.26	102.19	136.14	132.52	129.79
2	Basel	Switzerland	130.93	49.38	92.70	137.07	130.95	111.53

==========

We separated the City column into two different column as city and country. We drop the City column and Rank column(since it had NAs only) We, in any case, checked the leading and trailing space in city and country column. We checked if there were any NAs in any other columns. We made sure that there were no lower case or upper case type of inputs in city or Country column. We moved the city and country column to the head of the data set. Now we can use it with other datasets to work on our research.

In [35]:
```
### Now I will save this cleaned data as csv file.

cost_of_living_df.to_csv('clean_cost_of_living_df.csv', index = False)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: