DCS 540 Data Preparation (DSC540-T301 2225-1)

Bellevue University

Project: Milestone 2

Author: Jake Meyer

Date: 04/24/2022

Project Criteria for Milestone 2:

Perform at least 5 data transformation and/or cleansing steps to your flat file data. The below examples are not required - they are just potential transformations you could do. If your data doesn't work for these scenarios, complete different transformations. You can do the same transformation multiple times if needed to clean your data. The goal is a clean dataset at the end of the milestone.

Examples:

- Replace Headers
- Format data into a more readable format
- Identify outliers and bad data
- Find duplicates
- Fix casing or inconsistent values
- Conduct Fuzzy Matching

Make sure you clearly label each transformation step (Step #1, Step #2, etc.) in your code and describe what it is doing in 1-2 sentences. You can submit a Jupyter Notebook or a PDF of your code. If you submit a .py file you need to also include a PDF or attachment of your results.

Import the necessary libraries

In [152…

```
'''
Import the necessary libraries to complete the following activities:
1) Read in the csv file(s) and pdf file data.
2) Perform data transformation and cleansing steps.
3) Show visualization plots when/if applicable.
'''
import numpy as np

import pandas as pd
from pandas import Series, DataFrame
```

```
# import tabula - may be needed for pdf files
# from tabula import read_pdf - may be needed for pdf files

import matplotlib.pyplot as plt

from scipy import stats
```

Read in the csv file(s) and store the data as a DataFrame

In [153…

```
'''
Read in the csv file(s) with the pd.read_csv()
There are 4 csv files that will be reviewed.
df1 will store 2019_metro_areas.csv data.
df2 will store 2021_metro_areas.csv data.
df3 will store 2019_state_ranks.csv data.
df4 will store 2021_state_ranks.csv data.
'''
df1 = pd.read_csv("2019_metro_areas.csv")
df2 = pd.read_csv("2021_metro_areas.csv")
df3 = pd.read_csv("2019_state_ranks.csv")
df4 = pd.read_csv("2021_state_ranks.csv")
```

In [153…

```
'''
2019_metro_areas.csv was originally part of Milestone 1.
File name was updated to include year.
Link to original csv file is provided below:
https://www.kaggle.com/datasets/pileatedperch/best-cities-for-data-scientists?select=st
Look at the first 5 rows of df1.
'''
df1.head(5)
```

Out[153…

	Metro Area	Est. 2019 Population	Average Annual Salary	ZORI 2020-06	Avg Annual Rainfall	Lowest Monthly Avg High Temp	Highest Monthly Avg High Temp	Violent Crime Rate	Property Crime Rate	U.S. News Overall Score	C
0	Seattle, WA	3,979,845	63,120	1,983	37.5	46.0	76.0	353.7	1,963.60	7.2	
1	Austin, TX	2,227,083	51,840	1,565	32.2	62.0	96.0	306.3	2,343.40	7.6	
2	Salt Lake City, UT	1,232,696	47,272	1,361	16.1	37.0	93.0	257.0	3,075.00	6.9	
3	Spokane, WA	568,521	47,320	1,126	16.6	32.0	83.0	322.1	4,471.70	6.5	
4	Minneapolis-St. Paul, MN	3,654,908	56,030	1,565	30.6	24.0	83.0	283.0	2,404.20	7.3	

In [153…

```
'''
2021_metro_areas.csv was mentioned in Milestone 1 plan for more up-to-date rankings.
CSV File was generated based on the rankings reported from the U.S. News Article below:
https://realestate.usnews.com/places/rankings/best-places-to-live
Cities and scores were captured from the article and stored in this csv file.
```

```
Look at the first 5 rows of df2.
'''
df2.head(5)
```

Out[153...

	2021 Rank	Metro Area	U.S. News Overall Score	Quality of Life	Value
0	1	Boulder, CO	7.6	8.2	6.0
1	2	Raleigh, NC	7.5	6.9	7.7
2	3	Huntsville, AL	7.4	7.0	8.7
3	4	Fayetteville (AR), AR	7.3	7.2	8.1
4	5	Austin, TX	7.3	7.0	6.5

In [153...

```
'''
2019_state_ranks.csv was originally part of Milestone 1.
File name was updated to include year.
Link to original csv file is provided below:
https://www.kaggle.com/datasets/pileatedperch/best-cities-for-data-scientists?select=st
Look at the first 5 rows of df3.
'''
df3.head(5)
```

Out[153...

	state	state_name	Overall Rank	Healthcare	Education	Economy	Infrastructure	Opportunity	Fiscal Stability
0	WA	Washington	1	4	4	3	2	19	22
1	NH	New Hampshire	2	16	5	13	31	1	10
2	MN	Minnesota	3	10	17	18	6	3	25
3	UT	Utah	4	9	10	2	3	24	5
4	VT	Vermont	5	11	8	29	28	10	19

In [153...

```
'''
2021_state_ranks.csv was not mentioned in Milestone 1 plan.
CSV File was generated using tabula in conjunction with the U.S. News Article below:
https://realestate.usnews.com/places/rankings/best-places-to-live
This data will reviewed to determine if it is beneficial.
The data would add more up-to-date rankings for the states.
Look at the first 5 rows of df4.
'''
df4.head(5)
```

Out[153...

	RANK	state_name	state	CARE	EDUCATION	ECONOMY	INFRASTRUCTURE	OPPORTUNITY	STABII
0	1	Washington	WA	8	4	4	3	25	
1	2	Minnesota	MN	16	17	15	9	2	
2	3	Utah	UT	11	10	1	5	30	

	RANK	state_name	state	CARE	EDUCATION	ECONOMY	INFRASTRUCTURE	OPPORTUNITY	STABII
3	4	New Hampshire	NH	13	13	11	34	3	
4	5	Idaho	ID	24	29	3	10	24	

In [153…

```
'''
Understand the shape of each df.
'''
print("df1 shape is {}\n"
      "df2 shape is {}\n"
      "df3 shape is {} \n"
      "df4 shape is {}".format(df1.shape,df2.shape,df3.shape,df4.shape))
```

```
df1 shape is (125, 18)
df2 shape is (150, 5)
df3 shape is (50, 11)
df4 shape is (50, 11)
```

Perform data transformation and cleansing steps

Data Transformation and/or Cleansing Step 1:

Identify the critical columns from df1 and put them into the df_final DataFrame. Columns included in the DataFrame were chosen based on scope of the project for exploring the top cities to live in the U.S. along with useful insights for Data Scientist job outlooks within these recommended locations. This will be the starting point for creating the cleaned DataFrame df_final.

In [153…

```
'''
Data Transformation and/or Cleansing Step 1.
Specify the df1 columns of data to include in df_final.
'''
df1_names = ['Metro Area', 'Est. 2019 Population', 'Average Annual Salary',
             'Avg Annual Rainfall', 'Lowest Monthly Avg High Temp',
             'Highest Monthly Avg High Temp', 'Violent Crime Rate', 'Property Crime Rate
             'U.S. News Overall Score','U.S. News State Overall Rank', 'Data Scientist J
             '2019_rank']
df_final = df1[df1_names]
```

In [153…

```
'''
Look at the first 10 rows of the df_final DataFrame.
'''
df_final.head(10)
```

Out[153…

	Metro Area	Est. 2019 Population	Average Annual Salary	Avg Annual Rainfall	Lowest Monthly Avg High Temp	Highest Monthly Avg High Temp	Violent Crime Rate	Property Crime Rate	U.S. News Overall Score	U.S. News State Overall Rank
0	Seattle, WA	3,979,845	63,120	37.5	46.0	76.0	353.7	1,963.60	7.2	1.0

	Metro Area	Est. 2019 Population	Average Annual Salary	Avg Annual Rainfall	Lowest Monthly Avg High Temp	Highest Monthly Avg High Temp	Violent Crime Rate	Property Crime Rate	U.S. News Overall Score	U.S. News State Overall Rank
1	Austin, TX	2,227,083	51,840	32.2	62.0	96.0	306.3	2,343.40	7.6	38.0
2	Salt Lake City, UT	1,232,696	47,272	16.1	37.0	93.0	257.0	3,075.00	6.9	4.0
3	Spokane, WA	568,521	47,320	16.6	32.0	83.0	322.1	4,471.70	6.5	1.0
4	Minneapolis-St. Paul, MN	3,654,908	56,030	30.6	24.0	83.0	283.0	2,404.20	7.3	3.0
5	Denver, CO	2,967,239	57,400	14.3	43.0	89.0	413.9	1,600.90	7.4	10.0
6	Colorado Springs, CO	745,791	50,050	16.5	42.0	85.0	431.8	2,612.30	7.4	10.0
7	Boston, MA	4,873,019	65,420	43.8	36.0	81.0	305.3	1,291.00	6.9	8.0
8	Des Moines, IA	699,292	50,600	36.0	31.0	86.0	370.8	2,452.30	7.3	14.0
9	Boise, ID	749,202	43,880	11.7	38.0	91.0	235.0	1,761.90	7.1	16.0

Data Transformation and/or Cleansing Step 2:

Create 'city' and 'state' columns based from the 'Metro Area" column for the final DataFrame. This will entail splitting the strings from one column currently listed as city,state. This step is required so the data can be merged based on 'city' in a later step.

In [153…

```
'''
Data Transformation and/or Cleansing Step 2:
Split the 'Metro Area' column out by 'city' and 'state' for df_final.
Use the Series.str.split() function since the data is separated by a ','.
'''
df_final[['city','state']] = df_final['Metro Area'].str.split(",", n =1, expand = True)
```

```
C:\Users\jkmey\anaconda3\lib\site-packages\pandas\core\frame.py:3641: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
guide/indexing.html#returning-a-view-versus-a-copy
  self[k1] = value[k2]
```

In [153…

```
'''
Review first 5 rows of the df_final DataFrame to ensure 'city' and 'state' columns were
'''
df_final.head(5)
```

Out[153…

	Metro Area	Est. 2019 Population	Average Annual Salary	Avg Annual Rainfall	Lowest Monthly Avg High Temp	Highest Monthly Avg High Temp	Violent Crime Rate	Property Crime Rate	U.S. News Overall Score	U.S. News State Overall Rank
0	Seattle, WA	3,979,845	63,120	37.5	46.0	76.0	353.7	1,963.60	7.2	1.0
1	Austin, TX	2,227,083	51,840	32.2	62.0	96.0	306.3	2,343.40	7.6	38.0
2	Salt Lake City, UT	1,232,696	47,272	16.1	37.0	93.0	257.0	3,075.00	6.9	4.0
3	Spokane, WA	568,521	47,320	16.6	32.0	83.0	322.1	4,471.70	6.5	1.0
4	Minneapolis-St. Paul, MN	3,654,908	56,030	30.6	24.0	83.0	283.0	2,404.20	7.3	3.0

In [154…

```
'''
Move the 'city' and 'state' columns as the front columns for the df_final.
'''

df_final = df_final[['city', 'state', 'Metro Area', 'Est. 2019 Population', 'Average An
            'Avg Annual Rainfall', 'Lowest Monthly Avg High Temp',
            'Highest Monthly Avg High Temp', 'Violent Crime Rate', 'Property Crime Rate
            'U.S. News Overall Score','U.S. News State Overall Rank', 'Data Scientist J
            '2019_rank']]
```

Data Transformation and/or Cleansing Step 3:

Drop the 'Metro Area' column since it is no longer needed after splitting the city and state strings in the previous step. Additional columns will need to be dropped from df_final later, however the intent is to try to keep the df_final organized/clean through the process.

In [154…

```
'''
Data Transformation and/or Cleansing Step 3:
Use .drop() method on the "Metro Area" column in df_final.
'''
df_final = df_final.drop('Metro Area', axis = 1)
```

In [154…

```
'''
Review first 5 rows of df_final DataFrame to ensure columns are in the correct order an
'''
df_final.head(5)
```

Out[154…

	city	state	Est. 2019 Population	Average Annual Salary	Avg Annual Rainfall	Lowest Monthly Avg High Temp	Highest Monthly Avg High Temp	Violent Crime Rate	Property Crime Rate	U.S. News Overall Score	O
0	Seattle	WA	3,979,845	63,120	37.5	46.0	76.0	353.7	1,963.60	7.2	
1	Austin	TX	2,227,083	51,840	32.2	62.0	96.0	306.3	2,343.40	7.6	

	city	state	Est. 2019 Population	Average Annual Salary	Avg Annual Rainfall	Lowest Monthly Avg High Temp	Highest Monthly Avg High Temp	Violent Crime Rate	Property Crime Rate	U.S. News Overall Score	O
2	Salt Lake City	UT	1,232,696	47,272	16.1	37.0	93.0	257.0	3,075.00	6.9	
3	Spokane	WA	568,521	47,320	16.6	32.0	83.0	322.1	4,471.70	6.5	
4	Minneapolis-St. Paul	MN	3,654,908	56,030	30.6	24.0	83.0	283.0	2,404.20	7.3	

Data Transformation and/or Cleansing Step 4:

For the final DataFrame, specify the column names that were included from the first data source. This will help ensure the columns were from the 2019 data csv file.The additional columns merged into the df_final DataFrame will be named later.

In [154...
```
'''
Data Transformation and/or Cleansing Step 4:
Rename the columns in the df_final DataFrame.
'''
df_final_columns = ['city', 'state', '2019_est_population', '2019_average_annual_salary
                    '2019_average_annual_rainfall', '2019_lowest_monthly_average_low_tem
                     '2019_highest_monthly_average_high_temp', '2019_violent_crime_rate'
                    '2019_us_news_overall_score', '2019_us_news_state_overall_rank',
                    '2019_data_scientist_job_postings', '2019_rank']

df_final.columns = df_final_columns
```

In [154...
```
'''
Review the first 5 rows of df_final DataFrame to ensure the column names were revised.
'''

df_final.head(10)
```

Out[154...

	city	state	2019_est_population	2019_average_annual_salary	2019_average_annual_rainfall	2(
0	Seattle	WA	3,979,845	63,120	37.5	
1	Austin	TX	2,227,083	51,840	32.2	
2	Salt Lake City	UT	1,232,696	47,272	16.1	
3	Spokane	WA	568,521	47,320	16.6	
4	Minneapolis-St. Paul	MN	3,654,908	56,030	30.6	
5	Denver	CO	2,967,239	57,400	14.3	
6	Colorado Springs	CO	745,791	50,050	16.5	

	city	state	2019_est_population	2019_average_annual_salary	2019_average_annual_rainfall	2(
7	Boston	MA	4,873,019	65,420	43.8	
8	Des Moines	IA	699,292	50,600	36.0	
9	Boise	ID	749,202	43,880	11.7	

In [154…

```
'''
Understand the shape of the df_final DataFrame up through this point.
'''

print('df_final has {} rows and {} columns at this point.'.format(df_final.shape[0], df
```

```
df_final has 125 rows and 13 columns at this point.
```

Data Transformation and/or Cleansing Step 5:

Eliminate the whitespace present in the final DataFrame 'city' and 'state' columns. This will help when trying to join the additional datasources together in the next steps. Initially, ran into issues with whitespace and could have included the argument when the csv files were read above, however this section will address the whitespace around the strings.

In [154…

```
'''
Data Transformation and/or Cleansing Step 5:
Strip the whitespace in the 'city' and 'state' columns within df_final.
Use .str.strip() to remove the whitespace.
'''
df_final['city'].str.strip()
df_final['state'].str.strip()

df_final.head(10)
```

Out[154…

	city	state	2019_est_population	2019_average_annual_salary	2019_average_annual_rainfall	2(
0	Seattle	WA	3,979,845	63,120	37.5	
1	Austin	TX	2,227,083	51,840	32.2	
2	Salt Lake City	UT	1,232,696	47,272	16.1	
3	Spokane	WA	568,521	47,320	16.6	
4	Minneapolis-St. Paul	MN	3,654,908	56,030	30.6	
5	Denver	CO	2,967,239	57,400	14.3	
6	Colorado Springs	CO	745,791	50,050	16.5	
7	Boston	MA	4,873,019	65,420	43.8	
8	Des Moines	IA	699,292	50,600	36.0	
9	Boise	ID	749,202	43,880	11.7	

Data Transformation and/or Cleansing Step 6:

Merge the data from the second source (df2) into the final DataFrame. This step will require some additional transformation/cleaning steps to be performed prior to merging the data. The 'city' and 'state' columns in df2 will need to be split from 'Metro Area' similar to the previous step performed on df_final. In addition, the essential columns will need to be specified for what will be merged into final DataFrame.

In [154…
```
'''
Data Transformation and/or Cleansing Step 6:
Combine the data from df2 into df_final.
df2 will have 'Metro Area' split similar to the df_final step.
Split the Metro Area column out by 'city' and 'state' for df2.
Use the Series.str.split() function since the data is separated by a ','.
'''
df2[['city','state']] = df2['Metro Area'].str.split(",", n =1, expand = True)
```

In [154…
```
'''
Review df2 to see if 'city' and 'state' were separated out into two additional columns.
'''
df2.head(5)
```

Out[154…

	2021 Rank	Metro Area	U.S. News Overall Score	Quality of Life	Value	city	state
0	1	Boulder, CO	7.6	8.2	6.0	Boulder	CO
1	2	Raleigh, NC	7.5	6.9	7.7	Raleigh	NC
2	3	Huntsville, AL	7.4	7.0	8.7	Huntsville	AL
3	4	Fayetteville (AR), AR	7.3	7.2	8.1	Fayetteville (AR)	AR
4	5	Austin, TX	7.3	7.0	6.5	Austin	TX

In [154…
```
'''
Rename the columns in the d2 DataFrame.
'''
df2_column_names = ['2021_rank','metro_area', '2021_us_news_overall_score',
             '2021_quality_of_life', '2021_value', 'city', 'state']

df2.columns = df2_column_names
```

In [155…
```
'''
Drop the non-essential columns and arrange the columns for df2.
Strip the whitespace from the 'city' and 'state' data in df2.
Show the results.
'''
df2 = df2[['city','state','2021_rank','2021_us_news_overall_score',
          '2021_quality_of_life','2021_value']]
```

```
df2['city'].str.strip()
df2['state'].str.strip()

df2.head(5)
```

Out[155...

	city	state	2021_rank	2021_us_news_overall_score	2021_quality_of_life	2021_value
0	Boulder	CO	1	7.6	8.2	6.0
1	Raleigh	NC	2	7.5	6.9	7.7
2	Huntsville	AL	3	7.4	7.0	8.7
3	Fayetteville (AR)	AR	4	7.3	7.2	8.1
4	Austin	TX	5	7.3	7.0	6.5

In [155...

```
'''
Join df2 with the df_final DataFrame.
Start with df_final, merge based on 'city' with df2 and utilize drop_duplicates().
Argument for how to join will be 'right' to include all the cities from the most
up-to-date city rankings data.
'''
df_final = pd.merge(df_final, df2, on = 'city', how = 'right').drop_duplicates()
```

In [155...

```
'''
Review the df_final DataFrame with df2 joined.
Needed to go back and ensure duplicate city names were unique.
For example, Portland (OR) vs. Portland (ME).
Reran code without any further issues to this point.
'''
df_final.head(10)
```

Out[155...

	city	state_x	2019_est_population	2019_average_annual_salary	2019_average_annual_rainfall	2(
0	Boulder	NaN	NaN	NaN	NaN	
1	Raleigh	NaN	NaN	NaN	NaN	
2	Huntsville	AL	NaN	NaN	NaN	
3	Fayetteville (AR)	AR	534,904	45,830	47.9	
4	Austin	TX	2,227,083	51,840	32.2	
5	Colorado Springs	CO	745,791	50,050	16.5	
6	Naples	NaN	NaN	NaN	NaN	
7	Portland (ME)	ME	NaN	NaN	NaN	
8	Sarasota	FL	836,995	42,680	53.0	
9	Portland (OR)	OR	2,492,412	55,330	36.0	

In [155…
```
'''
Drop the 'state_x' column and rename the 'state_y' column to 'state'.
'''
df_final = df_final.drop('state_x', axis = 1)
df_final.rename(columns = {'state_y':'state'}, inplace = True)
```

In [155…
```
df_final.head(10)
```

Out[155…

	city	2019_est_population	2019_average_annual_salary	2019_average_annual_rainfall	2019_lowes
0	Boulder	NaN	NaN	NaN	
1	Raleigh	NaN	NaN	NaN	
2	Huntsville	NaN	NaN	NaN	
3	Fayetteville (AR)	534,904	45,830	47.9	
4	Austin	2,227,083	51,840	32.2	
5	Colorado Springs	745,791	50,050	16.5	
6	Naples	NaN	NaN	NaN	
7	Portland (ME)	NaN	NaN	NaN	
8	Sarasota	836,995	42,680	53.0	
9	Portland (OR)	2,492,412	55,330	36.0	

Data Transformation and/or Cleansing Step 7:

Combine the data from the third and fourth data sources stored under df3 and df4. Determine if it may be beneficial to merge the consolidated table from df3 and df4 with the df_final data. This section will require some transformation and cleaning steps specific to each of the DataFrames.

In [155…
```
'''
Data Transformation and/or Cleansing Step 7:
Combine the data from df3 and df4 together.
Specify the column names for df3.
Show the results.
'''
df3_column_names = ['state','state_name','2019_us_news_state_overall_rank',
                    '2019_healthcare_rank','2019_education_rank',
                    '2019_economy_rank','2019_infrasturcture_rank',
                    '2019_opportunity_rank','2019_fiscal_stability_rank',
                    '2019_crime&corrections_rank','2019_natural_environment_rank']

df3.columns = df3_column_names
df3.head(5)
```

Out[155...

	state	state_name	2019_us_news_state_overall_rank	2019_healthcare_rank	2019_education_rank	2019_
0	WA	Washington	1	4	4	
1	NH	New Hampshire	2	16	5	
2	MN	Minnesota	3	10	17	
3	UT	Utah	4	9	10	
4	VT	Vermont	5	11	8	

In [155...

```
'''
Drop the non-essential columns and arrange the columns for df3.
Strip the whitespace from the 'state' and 'state_name' data in df3.
Show the results.
'''
df3 = df3[['state','state_name','2019_us_news_state_overall_rank']]

df3['state'].str.strip()
df3['state_name'].str.strip()

df3.head(5)
```

Out[155...

	state	state_name	2019_us_news_state_overall_rank
0	WA	Washington	1
1	NH	New Hampshire	2
2	MN	Minnesota	3
3	UT	Utah	4
4	VT	Vermont	5

In [155...

```
'''
Specify the column names for df4.
Show the results.
'''
df4_column_names = ['2021_us_news_state_overall_rank','state_name','state','2021_health
                    '2021_education_rank','2021_economy_rank','2021_infrasturcture_rank
                    '2021_opportunity_rank','2021_fiscal_stability_rank',
                    '2021_crime&corrections_rank','2021_natural_environment_rank']

df4.columns = df4_column_names
df4.head(5)
```

Out[155...

	2021_us_news_state_overall_rank	state_name	state	2021_healthcare_rank	2021_education_rank	2021_
0	1	Washington	WA	8	4	
1	2	Minnesota	MN	16	17	
2	3	Utah	UT	11	10	

	2021_us_news_state_overall_rank	state_name	state	2021_healthcare_rank	2021_education_rank	2021_
3	4	New Hampshire	NH	13	13	
4	5	Idaho	ID	24	29	

In [155…
```
'''
Drop the non-essential columns and arrange the columns for df4.
Strip the whitespace from the 'state_name' and 'state' data in df3.
Show the results.
'''
df4 = df4[['2021_us_news_state_overall_rank','state_name','state']]

df4['state_name'].str.strip()
df4['state'].str.strip()

df4.head(5)
```

Out[155…

	2021_us_news_state_overall_rank	state_name	state
0	1	Washington	WA
1	2	Minnesota	MN
2	3	Utah	UT
3	4	New Hampshire	NH
4	5	Idaho	ID

In [155…
```
'''
Join df3 with df4 DataFrame through merge() based on 'state'.
'''
df3_df4_merged = pd.merge(df3, df4, on = 'state', how = 'outer').drop_duplicates()
```

In [156…
```
'''
Select the columns of interest to be included with the merged DataFrame for the third
and fourth data sources. The 2019 US News State Overall Rank is already included in the
final DataFrame, so this column will be removed. In addition, the state_name_y is
redundant and can be removed.
'''
df3_df4_merged = df3_df4_merged[['state','state_name_x', '2021_us_news_state_overall_ra
```

In [156…
```
'''
Specify the column names for df3_df4_merged DataFrame.
'''
df3_df4_merged_column_names = ['state','state_name','2021_us_news_state_overall_rank']

df3_df4_merged.columns = df3_df4_merged_column_names
df3_df4_merged.head(5)
```

Out[156…

	state	state_name	2021_us_news_state_overall_rank

	state	state_name	2021_us_news_state_overall_rank
0	WA	Washington	1.0
1	NH	New Hampshire	4.0
2	MN	Minnesota	2.0
3	UT	Utah	3.0
4	VT	Vermont	11.0

Data Transformation and/or Cleansing Step 8:

Attempted to merge the final DataFrame and the DataFrame generated from the previous step for sources 3 and 4.

In [156…

```
'''
Data Transformation and/or Cleansing Step 8:
Combine the merged df3_df4_merged DataFrame with df_final.
'''
df_final_attempt = pd.merge(df_final,df3_df4_merged,on='state',how='left').drop_duplica

df_final_attempt.head(10)
```

Out[156…

	city	2019_est_population	2019_average_annual_salary	2019_average_annual_rainfall	2019_lowe
0	Boulder	NaN	NaN	NaN	
1	Raleigh	NaN	NaN	NaN	
2	Huntsville	NaN	NaN	NaN	
3	Fayetteville (AR)	534,904	45,830	47.9	
4	Austin	2,227,083	51,840	32.2	
5	Colorado Springs	745,791	50,050	16.5	
6	Naples	NaN	NaN	NaN	
7	Portland (ME)	NaN	NaN	NaN	
8	Sarasota	836,995	42,680	53.0	
9	Portland (OR)	2,492,412	55,330	36.0	

After hours of trying to get the df_final and df3_df4_merged DataFrames together based on 'state', only NaN returns would show up with the additional columns. I tried creating a dictionary of 'state' as the key and "state_name" as the values in an attempt to map() a new column. This was also unsuccessful and resulted in NaN returns. As a result, I decided to leave the state data from the last two csv files out of the df_final.

Data Transformation and/or Cleansing Step 9:

Arrange the final DataFrame so the columns are in a logical order. Evaluate the columns for outliers and remove any nonessential or duplicate columns if necessary.

In [156...]
```
'''
Data Transformation and/or Cleansing Step 9:
Review what the final DataFrame looks like by viewing the first 10 rows.
'''
df_final.head(10)
```

Out[156...]

	city	2019_est_population	2019_average_annual_salary	2019_average_annual_rainfall	2019_lowes
0	Boulder	NaN	NaN	NaN	
1	Raleigh	NaN	NaN	NaN	
2	Huntsville	NaN	NaN	NaN	
3	Fayetteville (AR)	534,904	45,830	47.9	
4	Austin	2,227,083	51,840	32.2	
5	Colorado Springs	745,791	50,050	16.5	
6	Naples	NaN	NaN	NaN	
7	Portland (ME)	NaN	NaN	NaN	
8	Sarasota	836,995	42,680	53.0	
9	Portland (OR)	2,492,412	55,330	36.0	

In [156...]
```
'''
Drop the non-essential columns and arrange the columns for df_final.
Show the results for the first 30 rows.
'''
essentials = ['city','state','2021_rank','2021_us_news_overall_score',
              '2021_quality_of_life','2021_value','2019_rank','2019_us_news_overall_scor
              '2019_us_news_state_overall_rank', '2019_data_scientist_job_postings',
              '2019_est_population', '2019_average_annual_salary',
               '2019_average_annual_rainfall','2019_lowest_monthly_average_low_temp',
               '2019_highest_monthly_average_high_temp', '2019_violent_crime_rate', '201

df_final = df_final[essentials]

df_final.head(30)
```

Out[156...]

	city	state	2021_rank	2021_us_news_overall_score	2021_quality_of_life	2021_value	2019_ra
0	Boulder	CO	1	7.6	8.2	6.0	Na
1	Raleigh	NC	2	7.5	6.9	7.7	Na

	city	state	2021_rank	2021_us_news_overall_score	2021_quality_of_life	2021_value	2019_ra
2	Huntsville	AL	3	7.4	7.0	8.7	108
3	Fayetteville (AR)	AR	4	7.3	7.2	8.1	21
4	Austin	TX	5	7.3	7.0	6.5	2
5	Colorado Springs	CO	6	7.3	6.7	6.4	7
6	Naples	FL	7	7.2	7.7	4.7	Na
7	Portland (ME)	ME	8	7.2	7.4	6.6	105
8	Sarasota	FL	9	7.2	7.3	5.5	39
9	Portland (OR)	OR	10	7.2	6.8	6.0	11
10	Boise	ID	11	7.2	7.3	7.1	10
11	Ann Arbor	MI	12	7.2	8.4	7.0	Na
12	Des Moines	IA	13	7.2	6.7	8.3	9
13	Denver	CO	14	7.1	6.7	6.4	6
14	San Francisco	CA	15	7.1	6.8	5.1	23
15	Madison	WI	16	7.1	7.6	7.0	13
16	Fort Collins	CO	17	7.1	7.4	6.0	Na
17	Melbourne	FL	18	7.1	7.5	6.2	14
18	Seattle	WA	19	7.1	6.5	5.8	1
19	Charlotte	NC	20	7.1	6.0	7.5	17
20	Green Bay	WI	21	7.1	7.3	8.2	Na
21	Jacksonville	FL	22	7.0	6.8	6.3	30
22	Salt Lake City	UT	23	7.0	7.1	7.3	3
23	Spartanburg	SC	24	7.0	6.5	7.8	Na
24	Omaha	NE	25	7.0	6.8	7.9	26
25	Pensacola	FL	26	7.0	6.6	6.6	48
26	Minneapolis	MN	27	7.0	6.8	7.6	Na
27	Washington	DC	28	6.9	6.8	6.5	82
28	Lincoln	NE	29	6.9	6.9	7.6	Na
29	Nashville	TN	30	6.9	6.0	7.2	33

In [156…

```
print("The final DataFrame shape is {} rows and {} columns after the merges.".format(df
```

```
The final DataFrame shape is 150 rows and 17 columns after the merges.
```

Data Transformation and/or Cleansing Step 10:

Review the df_final data contents to ensure the data types are desireable. If not, then convert the column of data to the desired datatype.

In [156…
```
'''
Data Transformation and/or Cleansing Step 10:
Use the info() method to retrieve data type information about each column.
In addition, this will help identify null values (and non-null values for the next step
'''
df_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 150 entries, 0 to 149
Data columns (total 17 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   city                                    150 non-null    object
 1   state                                   150 non-null    object
 2   2021_rank                               150 non-null    int64
 3   2021_us_news_overall_score              150 non-null    float64
 4   2021_quality_of_life                    150 non-null    float64
 5   2021_value                              150 non-null    float64
 6   2019_rank                               122 non-null    float64
 7   2019_us_news_overall_score              122 non-null    float64
 8   2019_us_news_state_overall_rank         120 non-null    float64
 9   2019_data_scientist_job_postings        122 non-null    float64
 10  2019_est_population                     52 non-null     object
 11  2019_average_annual_salary              52 non-null     object
 12  2019_average_annual_rainfall            49 non-null     float64
 13  2019_lowest_monthly_average_low_temp    49 non-null     float64
 14  2019_highest_monthly_average_high_temp  49 non-null     float64
 15  2019_violent_crime_rate                 54 non-null     float64
 16  2019_property_crime_rate                54 non-null     object
dtypes: float64(11), int64(1), object(5)
memory usage: 21.1+ KB
```

In [156…
```
'''
Convert the columns associated with rank to integer data types.
This will include '2019_rank', '2019_us_news_state_overall_rank', and '2019_data_scient
'''
df_final['2019_rank'] = pd.array(df_final['2019_rank'], dtype=pd.Int64Dtype())
df_final['2019_us_news_state_overall_rank'] = pd.array(df_final['2019_us_news_state_ove
df_final['2019_data_scientist_job_postings'] = pd.array(df_final['2019_data_scientist_j
```

In [156…
```
df_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 150 entries, 0 to 149
Data columns (total 17 columns):
 #   Column                                  Non-Null Count  Dtype
---  ------                                  --------------  -----
 0   city                                    150 non-null    object
```

```
 1   state                                 150 non-null    object
 2   2021_rank                             150 non-null    int64
 3   2021_us_news_overall_score            150 non-null    float64
 4   2021_quality_of_life                  150 non-null    float64
 5   2021_value                            150 non-null    float64
 6   2019_rank                             122 non-null    Int64
 7   2019_us_news_overall_score            122 non-null    float64
 8   2019_us_news_state_overall_rank       120 non-null    Int64
 9   2019_data_scientist_job_postings      122 non-null    Int64
 10  2019_est_population                   52 non-null     object
 11  2019_average_annual_salary            52 non-null     object
 12  2019_average_annual_rainfall          49 non-null     float64
 13  2019_lowest_monthly_average_low_temp  49 non-null     float64
 14  2019_highest_monthly_average_high_temp  49 non-null    float64
 15  2019_violent_crime_rate               54 non-null     float64
 16  2019_property_crime_rate              54 non-null     object
dtypes: Int64(3), float64(8), int64(1), object(5)
memory usage: 21.5+ KB
```

Data Transformation and/or Cleansing Step 11:

Review the df_final data contents for missing values and understand where the missing values are located. Determine the best outcome for handling the missing values.

In [156…

```
'''
From the previous output, we know which columns contain non-null values
and could figure out which columns will contain null values by default.
However, I will use the isnull() method to check for missing values within the df_final
'''
df_final.isnull()
```

Out[156…

	city	state	2021_rank	2021_us_news_overall_score	2021_quality_of_life	2021_value	2019_rank	2(
0	False	False	False	False	False	False	True	
1	False	False	False	False	False	False	True	
2	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	
145	False	False	False	False	False	False	False	
146	False	False	False	False	False	False	False	
147	False	False	False	False	False	False	False	
148	False	False	False	False	False	False	True	
149	False	False	False	False	False	False	False	

150 rows × 17 columns

In [157…

```
'''
```

```
Show which columns contain the NaN values with isnull() and isany().
'''
df_final.isnull().any()
```

Out[157…

```
city                                    False
state                                   False
2021_rank                               False
2021_us_news_overall_score              False
2021_quality_of_life                    False
2021_value                              False
2019_rank                                True
2019_us_news_overall_score               True
2019_us_news_state_overall_rank          True
2019_data_scientist_job_postings         True
2019_est_population                      True
2019_average_annual_salary               True
2019_average_annual_rainfall             True
2019_lowest_monthly_average_low_temp     True
2019_highest_monthly_average_high_temp   True
2019_violent_crime_rate                  True
2019_property_crime_rate                 True
dtype: bool
```

In [157…

```
'''
There is complete data for the 2021 columns and partial data for the 2019 columns.
Show the sum of missing values for each of the columns within the df_final DataFrame.
'''
df_final.isnull().sum()
```

Out[157…

```
city                                     0
state                                    0
2021_rank                                0
2021_us_news_overall_score               0
2021_quality_of_life                     0
2021_value                               0
2019_rank                               28
2019_us_news_overall_score              28
2019_us_news_state_overall_rank         30
2019_data_scientist_job_postings        28
2019_est_population                     98
2019_average_annual_salary              98
2019_average_annual_rainfall           101
2019_lowest_monthly_average_low_temp   101
2019_highest_monthly_average_high_temp 101
2019_violent_crime_rate                 96
2019_property_crime_rate                96
dtype: int64
```

In [157…

```
'''
The NaN values in '2019_rank' will be filled with a 0 because they were not on the list
The same concept will be applied for '2019_us_news_state_overall_rank','2019_us_news_ov
and '2019_data_scientist_job_postings'.
'''
df_final['2019_rank']=df_final['2019_rank'].fillna(0)
df_final['2019_us_news_overall_score']=df_final['2019_us_news_overall_score'].fillna(0)
df_final['2019_data_scientist_job_postings']=df_final['2019_data_scientist_job_postings
df_final['2019_us_news_state_overall_rank']=df_final['2019_us_news_state_overall_rank']
```

In [157…
```
'''
The NaN values were filled with a 0 for cities that were listed on the 2019 top cities,
for the 4 columns listed in the previous cell.
There are now 7 columns that have a significant amount of missing values as seen below:
'''
df_final.isnull().sum()
```

Out[157…
```
city                                  0
state                                 0
2021_rank                             0
2021_us_news_overall_score            0
2021_quality_of_life                  0
2021_value                            0
2019_rank                             0
2019_us_news_overall_score            0
2019_us_news_state_overall_rank       0
2019_data_scientist_job_postings      0
2019_est_population                  98
2019_average_annual_salary           98
2019_average_annual_rainfall        101
2019_lowest_monthly_average_low_temp 101
2019_highest_monthly_average_high_temp 101
2019_violent_crime_rate              96
2019_property_crime_rate             96
dtype: int64
```

In [157…
```
'''
With the amount of data missing from the remaining 7 columns.
As a result, I'm going to drop the remaining columns that have null values.
'''
df_final = df_final.dropna(axis=1)
```

In [157…
```
df_final.isnull().sum()
```

Out[157…
```
city                                0
state                               0
2021_rank                           0
2021_us_news_overall_score          0
2021_quality_of_life                0
2021_value                          0
2019_rank                           0
2019_us_news_overall_score          0
2019_us_news_state_overall_rank     0
2019_data_scientist_job_postings    0
dtype: int64
```

In [157…
```
'''
Show the final DataFrame after all the Transformation and/or Cleansing Steps.
'''
df_final
```

Out[157…

	city	state	2021_rank	2021_us_news_overall_score	2021_quality_of_life	2021_value	2019_ra
0	Boulder	CO	1	7.6	8.2	6.0	
1	Raleigh	NC	2	7.5	6.9	7.7	

	city	state	2021_rank	2021_us_news_overall_score	2021_quality_of_life	2021_value	2019_ra
2	Huntsville	AL	3	7.4	7.0	8.7	1
3	Fayetteville (AR)	AR	4	7.3	7.2	8.1	
4	Austin	TX	5	7.3	7.0	6.5	
...	...	...	...	...	...	...	
145	Modesto	CA	146	5.5	5.8	4.8	
146	Stockton	CA	147	5.4	5.1	4.8	
147	Bakersfield	CA	148	5.4	5.2	5.1	
148	Visalia	CA	149	5.2	6.4	4.5	
149	San Juan	PR	150	3.8	2.8	2.5	1

150 rows × 10 columns

In [157...]
```
print('The shape of the final DataFrame is {} rows and {} columns.'.format(df_final.sha
```

```
The shape of the final DataFrame is 150 rows and 10 columns.
```

In [152...]
```
'''
Export df_final to a csv file.
Label the file as df_final.csv
'''
df_final.to_csv('df_final.csv',index=False)
```

Ethical Considerations for Milestone 2:

The steps outlined above were followed to arrive at the df_final DataFrame. The first portion of this ethical section will focus on the source data. This DataFrame consists of 2021 and 2019 data from US News World Report "Best Places to Live". The 2019_metro_areas.csv data was available from Kaggle (Joe Corliss, 2020). The 2021_metro_areas.csv file was compiled from the US News World Report site with the most up-to-date rankings. Two additional files (2019_state_ranks.csv and 2021_state_rank.csv) were reviewed in this project as well. These were pulled from the same two sources for state rankings rather than city. The state CSV data was not included in the final DataFrame. When trobuleshooting for merging the state CSV with the Metro Area data, I did change the column name for 2019_state_ranks.csv to "state" and "state_name". The same changes were made in 2021_state_ranks.csv for the specific column headers, however I added a column with the State abbreviations. The next ethical topic will be the handling of missing values. There were 11 columns in the final DataFrame that contained missing values. Four of those columns had the missing values filled with a "0" to symbolize the data was not available since the city was not on the top list for 2021 (and it was for 2022). The columns which had these missing values filled were "2019_rank", "2019_us_news_state_overall_rank","2019_us_news_overall_score", and "2019_data_scientist_job_postings". The remaining seven columns all had over 75% of the values

missing for each respective column. As a result, these columns were dropped from df_final DataFrame. The steps performed to arrive at this final dataset are outlined for transparency.