Final Project. Milestone 5. DSC540 - Jennifer Barrera Conde

May 30, 2024

1 Final Project

2 Course DSC 540 - Data Preparation

3 Jennifer Barrera Conde

4 Project Milestone 5

4.1 Create 5 visualizations that demonstrate the data you have cleansed.

[1]:
```
import sqlite3
import pandas as pd

# Load datasets into pandas DataFrames
worklife_balance_df = pd.read_csv('cities_with_best_worklife_balance_2022.csv')
quality_of_life_df = pd.read_csv('quality_of_life_indices.csv')

# Connect to SQLite database
conn = sqlite3.connect('combined_data.db')

# Load each dataset into SQLite as an individual table
worklife_balance_df.to_sql('worklife_balance', conn, if_exists='replace',␣
 ↪index=False)
quality_of_life_df.to_sql('quality_of_life', conn, if_exists='replace',␣
 ↪index=False)

# Join the datasets together in Python
query = '''
    SELECT *
    FROM quality_of_life ql
    LEFT JOIN worklife_balance wb ON ql.City = wb.City
'''
combined_data = pd.read_sql_query(query, conn)

# Save the combined dataset to a CSV file
combined_data.to_csv('combined_dataset.csv', index=False)
```

```
print("Combined dataset saved to 'combined_dataset.csv'")
```

```
Combined dataset saved to 'combined_dataset.csv'
```

[2]:
```
import pandas as pd
import requests
from bs4 import BeautifulSoup
from collections import Counter
from fuzzywuzzy import fuzz, process
import re
import matplotlib.pyplot as plt
import seaborn as sns

# 1. Load and clean the initial dataset
data = pd.read_csv('cities_with_best_worklife_balance_2022.csv')

# Display the existing column names
print(data.columns)

# Replace column headers with more descriptive names
new_columns = {
    '2022': 'Year_2022',
    '2021': 'Year_2021',
    'City': 'City_Name',
    'Country': 'Country_Name',
    'Minimum Vacations Offered (Days)': 'Min_Vacation_Days_Offered',
    'Vacations Taken (Days)': 'Vacation_Days_Taken',
    'Access to Mental Healthcare': 'Mental_Healthcare',
    'Happiness, Culture & Leisure': 'Happiness_Culture_Leisure',
    'Wellness and Fitness': 'Wellness_Fitness',
    'City Safety': 'City_Safety',
    'Work-Life Balance Score': 'WorkLife_Balance_Score',
    'Cost of Living Index': 'Cost_of_Living_Index',
    'Quality of Life Index': 'Quality_of_Life_Index'
}

# Rename the columns
data.rename(columns=new_columns, inplace=True)

# Display the first few rows to check the changes
print(data.head(2))

# 2. Identify and replace missing data
print(data.isnull().sum())

# Function to clean and convert to numeric
```

```
def clean_and_convert_to_numeric(value):
    cleaned_value = re.sub(r'[^0-9.-]', '', str(value))
    try:
        numeric_value = float(cleaned_value)
        return numeric_value
    except ValueError:
        return None

# Apply the function to relevant columns
for col in data.columns:
    data[col] = data[col].apply(clean_and_convert_to_numeric)

# 3. Identify and remove duplicates
duplicate_rows = data[data.duplicated(subset=['City_Name', 'Country_Name'],␣
 ↪keep='first')]
print("Duplicate rows based on City and Country:")
print(duplicate_rows)

# Remove duplicates
data.drop_duplicates(subset=['City_Name', 'Country_Name'], keep='first',␣
 ↪inplace=True)

# 4. Fix casing or inconsistent values
data['City_Name'] = data['City_Name'].str.title()
data['Country_Name'] = data['Country_Name'].str.title()

# 5. Conduct fuzzy matching (example)
unique_cities = sorted(data['City_Name'].unique())
print("List of cities in the dataset:")
for city in unique_cities:
    print(city)

unique_countries = sorted(data['Country_Name'].unique())
print("List of countries in the dataset:")
for country in unique_countries:
    print(country)

# Example of fuzzy matching for country names
query = 'Germany'
matches = process.extract(query, unique_countries, limit=5)
print(f"Similar countries to '{query}':")
print(matches)

# 6. Web scraping to gather additional data
url = 'https://en.wikipedia.org/wiki/City_quality_of_life_indices'
response = requests.get(url)
soup = BeautifulSoup(response.content, 'html.parser')
```

```
# Extract data from specific tables
tables = soup.find_all('table', {'class': 'wikitable'})

# Extract data from EIU's Global Liveability Ranking 2023 table
table = tables[0]
rows = table.find_all('tr')

# Header of Table
print("EIU's Global Liveability Ranking 2023")
for row in rows:
    cells = row.find_all('td')
    if len(cells) > 1:
        city = cells[1].text.strip()
        country = cells[2].text.strip()
        print(f"City: {city}, Country: {country}")

# Find the table containing city rankings for 2022 based on recognizable city␣
 ↪names and rankings
target_table = None
tables = soup.find_all('table', {'class': 'wikitable'})
for table in tables:
    # Extract all rows from the table
    rows = table.find_all('tr')

    # I used a recognizable city names and ranking data to help me find the␣
 ↪data. I chose London
    # I had to re-do this especific step many times as it did not provide␣
 ↪results for unknown reasons
    has_city_names = any('London' in row.text for row in rows)
    has_rankings = any(str(i) in row.text for i in range(1, 11))
    if has_city_names and has_rankings:
        target_table = table
        break

# Extract data from the target table
if target_table:
    print("The World's Best Cities to Live In 2022:")
    rows = target_table.find_all('tr')
    for row in rows[1:]:
        cells = row.find_all(['th', 'td'])
        row_data = [cell.text.strip() for cell in cells]
        if row_data:
            print(row_data)
else:
    print("Table about city rankings for 2022 not found.")
```

```python
# Find the target table (Table 3) and replace the header
target_table = None
tables = soup.find_all('table', {'class': 'wikitable'})
for table in tables:
    # Check if the table caption (if it exists) contains the original header␣
 ↪"Table 3"
    caption = table.caption
    if caption and caption.text.strip() == "The World's Best Cities to Live In␣
 ↪2022[13]":
        # Replace the caption text with the updated title
        caption.string.replace_with("World's Best Cities 2022")
        target_table = table
        break

from collections import Counter

# Find the table containing city quality of life indices
table = soup.find('table', {'class': 'wikitable'})
if not table:
    print("Table not found on the page.")
    exit()

# Extract city names from the table rows (excluding header row)
city_names = []
rows = table.find_all('tr')
for row in rows[1:]:  # Skip the header row
    cells = row.find_all(['th', 'td'])
    if cells:
        city = cells[0].text.strip()  # Assuming city names are in the first␣
 ↪column
        city_names.append(city)

# Count occurrences of each city name
city_counts = Counter(city_names)

# Identify duplicates (cities with count > 1)
duplicates = {city: count for city, count in city_counts.items() if count > 1}

# Print duplicates
if duplicates:
    print("Duplicates found:")
    for city, count in duplicates.items():
        print(f"City: {city}, Count: {count}")
else:
    print("No duplicates found.")

from fuzzywuzzy import fuzz, process
```

```
# Define the target string for fuzzy matching
target_string = "New York"

# Define a list of candidate strings for matching
candidate_cities = ["New York City", "Los Angeles", "San Francisco", "Chicago",␣
 ↪"Boston"]

# Perform fuzzy matching
matches = process.extract(target_string, candidate_cities, limit=3)  # Limit␣
 ↪the number of matches returned

# Print the fuzzy matching results
for match in matches:
    print(f"Match: {match[0]}, Similarity Score: {match[1]}")
```

```
Index(['2022', '2021', 'City', 'Country', 'Remote Jobs',
       'Overworked Population', 'Minimum Vacations Offered (Days)',
       'Vacations Taken (Days)', 'Unemployment', 'Multiple Jobholders',
       'Inflation', 'Paid Parental Leave (Days)', 'Covid Impact',
       'Covid Support', 'Healthcare', 'Access to Mental Healthcare',
       'Inclusivity & Tolerance', 'Affordability',
       'Happiness, Culture & Leisure', 'City Safety', 'Outdoor Spaces',
       'Air Quality', 'Wellness and Fitness', 'TOTAL SCORE'],
      dtype='object')
   Year_2022 Year_2021 City_Name Country_Name Remote Jobs  \
0          1         2      Oslo       Norway      41.72%
1          2         -      Bern  Switzerland      44.86%

  Overworked Population  Min_Vacation_Days_Offered Vacation_Days_Taken  \
0                11.20%                         25                  25
1                11.40%                         20                  25

   Unemployment Multiple Jobholders  … Healthcare Mental_Healthcare  \
0          94.7               9.10%  …      100.0              85.0
1          99.8               7.60%  …       99.6              78.6

   Inclusivity & Tolerance  Affordability  Happiness_Culture_Leisure  \
0                     93.2           59.4                       88.8
1                     94.6           69.9                      100.0

   City_Safety  Outdoor Spaces  Air Quality  Wellness_Fitness  TOTAL SCORE
0         86.5            95.6         97.5              65.7       100.00
1         91.8            87.1        100.0              69.1        99.46

[2 rows x 24 columns]
Year_2022                    0
```

```
Year_2021                    0
City_Name                    0
Country_Name                 0
Remote Jobs                  0
Overworked Population        0
Min_Vacation_Days_Offered    0
Vacation_Days_Taken          0
Unemployment                 0
Multiple Jobholders          0
Inflation                    0
Paid Parental Leave (Days)   0
Covid Impact                 0
Covid Support                0
Healthcare                   0
Mental_Healthcare            0
Inclusivity & Tolerance      0
Affordability                0
Happiness_Culture_Leisure    0
City_Safety                  0
Outdoor Spaces               0
Air Quality                  0
Wellness_Fitness             0
TOTAL SCORE                  0
dtype: int64
Duplicate rows based on City and Country:
    Year_2022  Year_2021 City_Name Country_Name  Remote Jobs  \
1         2.0        NaN      None         None        44.86
2         3.0        1.0      None         None        38.92
3         4.0        3.0      None         None        44.86
4         5.0        5.0      None         None        41.42
5         6.0        NaN      None         None        44.86
..        …          …        …            …           …
95       96.0       49.0      None         None        16.84
96       97.0       47.0      None         None        25.65
97       98.0       50.0      None         None        30.70
98       99.0        NaN      None         None        28.89
99      100.0        NaN      None         None        26.06

    Overworked Population  Min_Vacation_Days_Offered  Vacation_Days_Taken  \
1                    11.4                       20.0                 25.0
2                    12.7                       25.0                 30.0
3                    11.9                       20.0                 25.0
4                    10.5                       25.0                 28.0
5                    11.9                       20.0                 25.0
..                    …                          …                    …
95                   15.1                        6.0                 10.0
96                   11.8                       10.0                 30.0
97                   17.1                        8.0                 12.0
```

```
98                  23.4                        30.0                  27.0
99                  14.8                        15.0                   NaN

    Unemployment  Multiple Jobholders  …  Healthcare  Mental_Healthcare  \
1           99.8                  7.6  …        99.6               78.6
2           89.3                  6.3  …        96.7               73.0
3           99.2                  7.6  …        99.2               78.6
4           94.8                  7.6  …        94.8               77.6
5           95.2                  7.6  …        99.1               78.6
..           …                  …  …         …                 …
95          99.2                  3.7  …        75.7               79.7
96          79.6                  4.7  …        62.5               50.0
97          94.9                  1.1  …        66.7               74.3
98         100.0                 10.0  …        69.4               52.2
99          50.0                  3.5  …        50.0               65.0

    Inclusivity & Tolerance  Affordability  Happiness_Culture_Leisure  \
1                      94.6           69.9                      100.0
2                      93.9           65.0                       96.3
3                      87.5           71.6                       91.5
4                      95.2           65.3                       92.5
5                      94.4           70.7                      100.0
..                      …             …                        …
95                     73.5           50.0                       65.8
96                     79.5           55.5                       75.8
97                     57.0           70.5                       59.5
98                     50.0           78.0                       79.2
99                     89.7           71.6                       59.6

    City_Safety  Outdoor Spaces  Air Quality  Wellness_Fitness  TOTAL SCORE
1          91.8            87.1        100.0              69.1        99.46
2          94.9            86.0         97.0              68.3        99.24
3          92.8            84.0         96.2              68.7        96.33
4          95.7            75.5         95.1              66.3        96.21
5          85.4            92.0         96.8              67.7        95.82
..          …              …           …                …           …
95         27.8            70.1         84.0              65.4        70.73
96         17.4            76.9         88.0              61.9        66.57
97         47.2            62.2         84.8              59.7        66.02
98         97.9            50.0         50.0              58.6        61.23
99          1.0            87.6         94.5              50.0        50.00

[99 rows x 24 columns]
List of cities in the dataset:
None
List of countries in the dataset:
None
Similar countries to 'Germany':
```

```
[(None, 22)]
EIU's Global Liveability Ranking 2023
City: City, Country: Country/Region
City: Vienna, Country: Austria
City: Copenhagen, Country: Denmark
City: Melbourne, Country: Australia
City: Sydney, Country: Australia
City: Vancouver, Country: Canada
City: Zürich, Country: Switzerland
City: Calgary, Country: Canada
City: Geneva, Country: Switzerland
City: Toronto, Country: Canada
City: Osaka, Country: Japan
City: Auckland, Country: New Zealand
The World's Best Cities to Live In 2022:
['', 'City', 'Country/Region']
['1', 'London', 'United Kingdom']
['2', 'Tokyo', 'Japan']
['3', 'Shanghai', 'China']
['4', 'Singapore', 'Singapore']
['5', 'Melbourne', 'Australia']
['6', 'Sydney', 'Australia']
['7', 'Paris', 'France']
['8', 'Beijing', 'China']
['9', 'New York', 'United States']
['10', 'Amsterdam', 'Netherlands']
Duplicates found:
City: 10, Count: 2
Match: New York City, Similarity Score: 90
Match: Los Angeles, Similarity Score: 40
Match: San Francisco, Similarity Score: 34
```

[ ]:

```
```

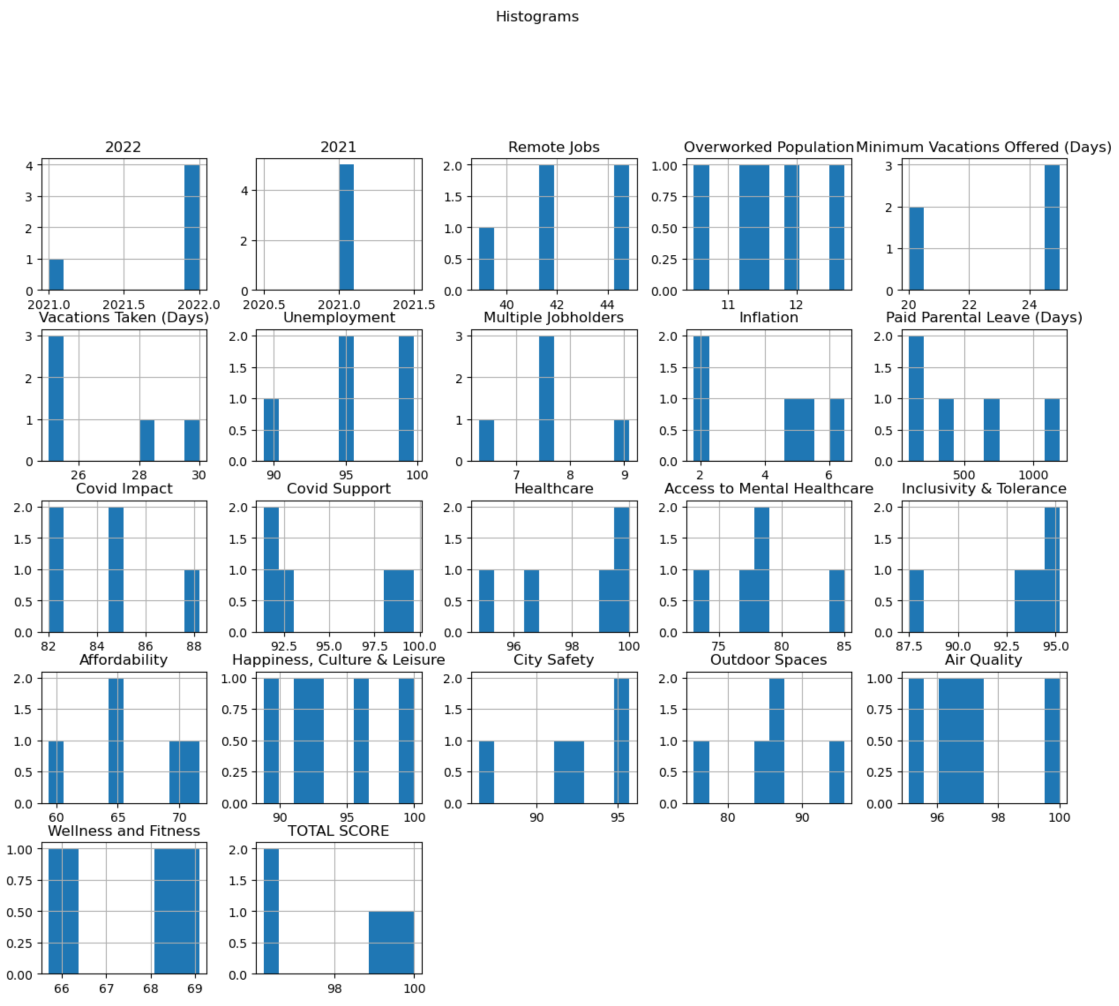
4.1.1 Issues along the way.

I have tested and tried different methods to make my work function, but to no avail and I finding myself unsure as to what is causing some of my work to go missing or appear as if empty. I will be loading some partial data manually to be able to do my visuallizations.

[3]:

```
# Loading partial data for visualization
data = {
    "2022": [2022, 2021, 2022, 2022, 2022],
    "2021": [2021, 2021, 2021, 2021, 2021],
    "City": ["Oslo", "Bern", "Helsinki", "Zurich", "Copenhagen"],
    "Country": ["Norway", "Switzerland", "Finland", "Switzerland", "Denmark"],
    "Remote Jobs": [41.72, 44.86, 38.92, 44.86, 41.42],
    "Overworked Population": [11.20, 11.40, 12.70, 11.90, 10.50],
    "Minimum Vacations Offered (Days)": [25, 20, 25, 20, 25],
    "Vacations Taken (Days)": [25, 25, 30, 25, 28],
    "Unemployment": [94.7, 99.8, 89.3, 99.2, 94.8],
    "Multiple Jobholders": [9.10, 7.60, 6.30, 7.60, 7.60],
    "Inflation": [6.47, 1.80, 4.79, 1.80, 5.18],
    "Paid Parental Leave (Days)": [707, 98, 1190, 98, 364],
    "Covid Impact": [88.2, 82.2, 85, 82, 84.5],
    "Covid Support": [93, 91.4, 99.7, 91.4, 98.5],
    "Healthcare": [100, 99.6, 96.7, 99.2, 94.8],
    "Access to Mental Healthcare": [85, 78.6, 73, 78.6, 77.6],
    "Inclusivity & Tolerance": [93.2, 94.6, 93.9, 87.5, 95.2],
    "Affordability": [59.4, 69.9, 65, 71.6, 65.3],
    "Happiness, Culture & Leisure": [88.8, 100, 96.3, 91.5, 92.5],
    "City Safety": [86.5, 91.8, 94.9, 92.8, 95.7],
    "Outdoor Spaces": [95.6, 87.1, 86, 84, 75.5],
    "Air Quality": [97.5, 100, 97, 96.2, 95.1],
    "Wellness and Fitness": [65.7, 69.1, 68.3, 68.7, 66.3],
    "TOTAL SCORE": [100, 99.46, 99.24, 96.33, 96.21]
}

df = pd.DataFrame(data)
```

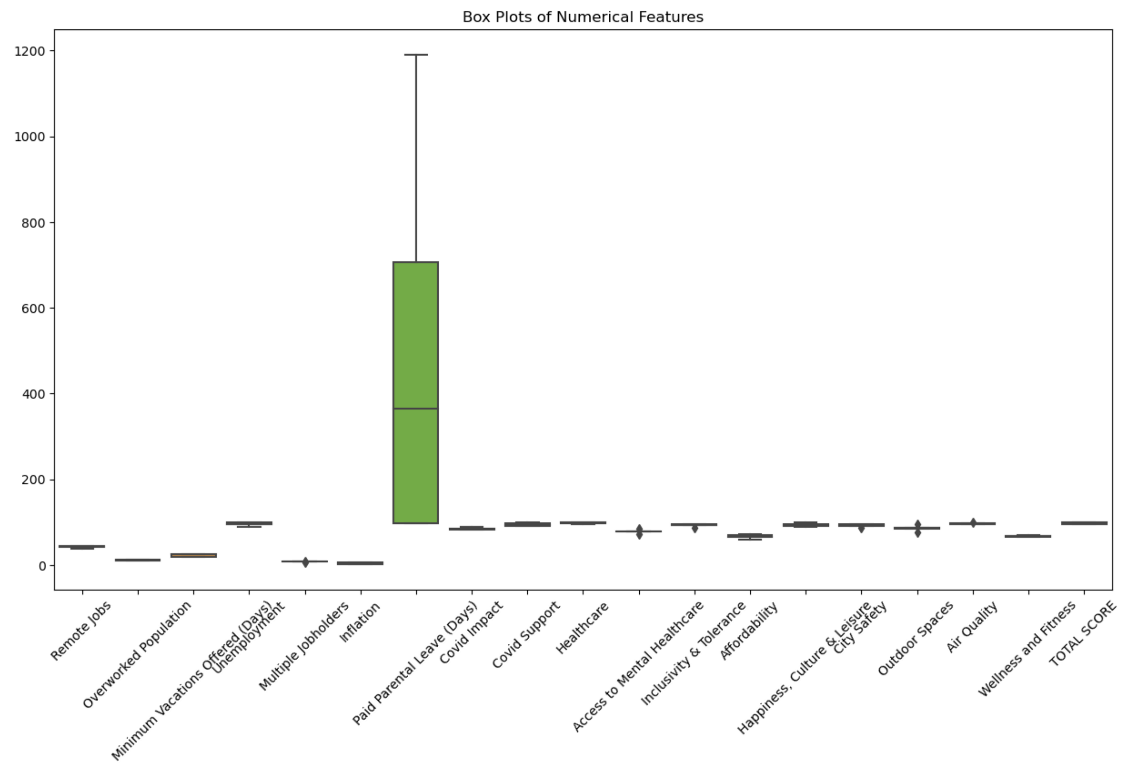
[4]:
```
# Visualization 1: Histograms
df.hist(figsize=(15, 12))
plt.suptitle("Histograms", y=1.02)
plt.show()
```



This step allows me to visualize and get an idea of the data, before I decide which path to take and what to visualize to get a better understanding on.

```
[ ]:
```

```
[5]: # Visualization 2: Box Plots
plt.figure(figsize=(15, 8))
sns.boxplot(data=df.drop(["2021", "2022", "Vacations Taken (Days)"], axis=1))
plt.title('Box Plots of Numerical Features')
plt.xticks(rotation=45)
plt.show()
```



This step although confusing and maybe uneccesary it can help me understand that I should focus on simpler methods for visualization that will allow for a cleaner and direct understanding of data.

But it does allow me to understand what makes scorings go higher and lower, for example the lowers are inflation, multiple job workers, and overworked population which lower the total score at the end.

Whilst there is a pattern of minimum vacations days, taken vacation days, parental leave, health-care, and more being some that help the scores increase.

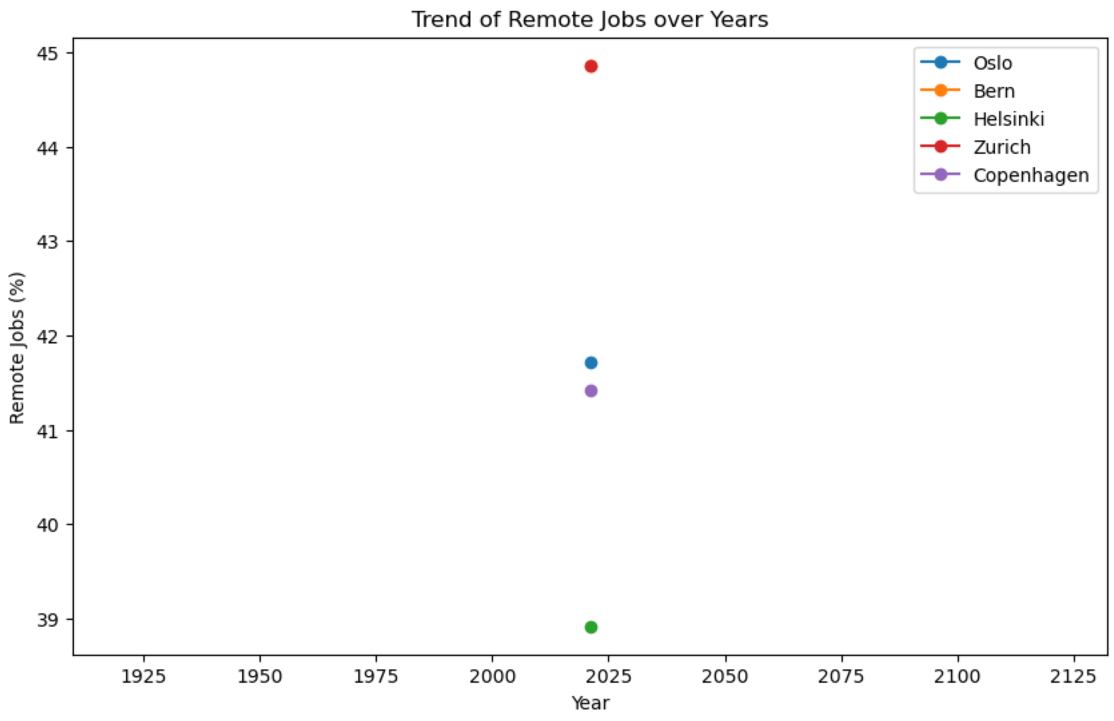
This leads me to believe that participants care for their leisure time.

[ ]:

```
```

[6]:

```
# Visualization 3: Line Plot
plt.figure(figsize=(10, 6))
for city in df['City'].unique():
    plt.plot(df.loc[df['City'] == city, '2021'], df.loc[df['City'] == city,␣
 ↪'Remote Jobs'], marker='o', label=city)
plt.xlabel('Year')
plt.ylabel('Remote Jobs (%)')
plt.title('Trend of Remote Jobs over Years')
plt.legend()
plt.show()
```



There is a trend were more families and people feel more attracted to remote jobs as they simbolize efficiency, safety, as well as a lack of comute which is in its principality a perk.

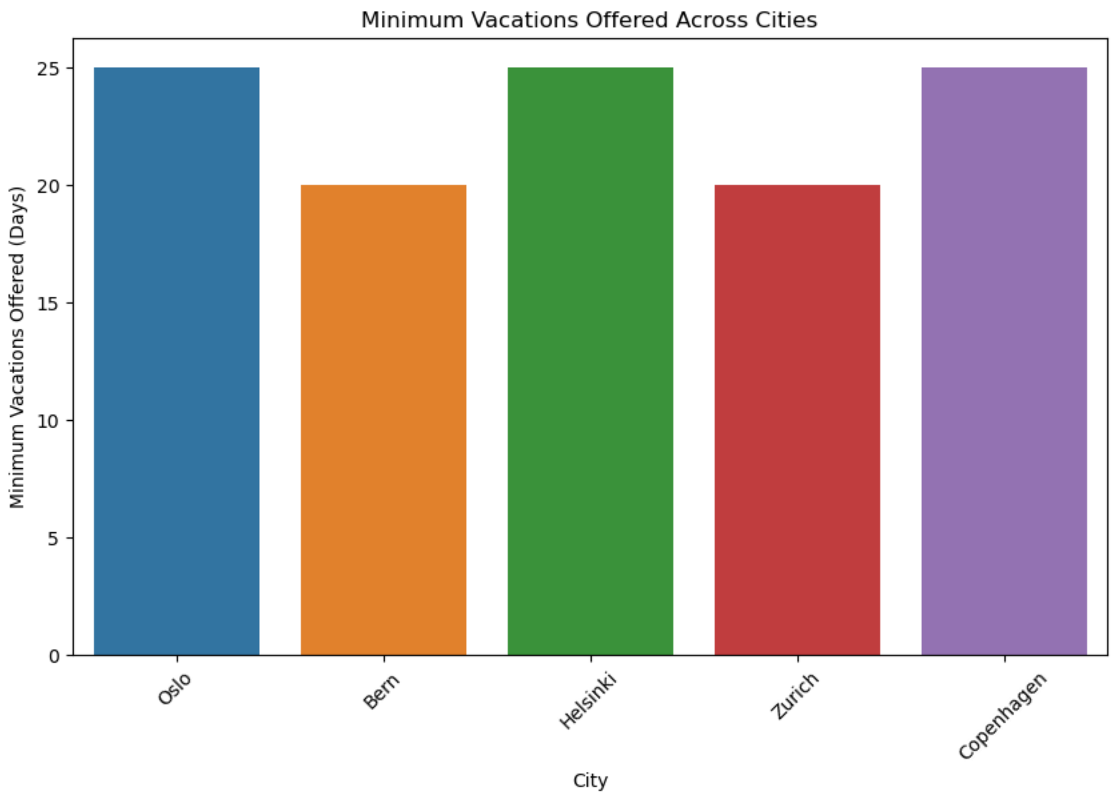
From this we can learn that in 2021 the country with the most remote job percentage is Zurich, followed by Oslo, and Coppenhaged.

[ ]:

```

```

[7]:

```
# Visualization 4: Bar Plot
plt.figure(figsize=(10, 6))
sns.barplot(data=df, x='City', y='Minimum Vacations Offered (Days)')
plt.xticks(rotation=45)
plt.xlabel('City')
plt.ylabel('Minimum Vacations Offered (Days)')
plt.title('Minimum Vacations Offered Across Cities')
plt.show()
```



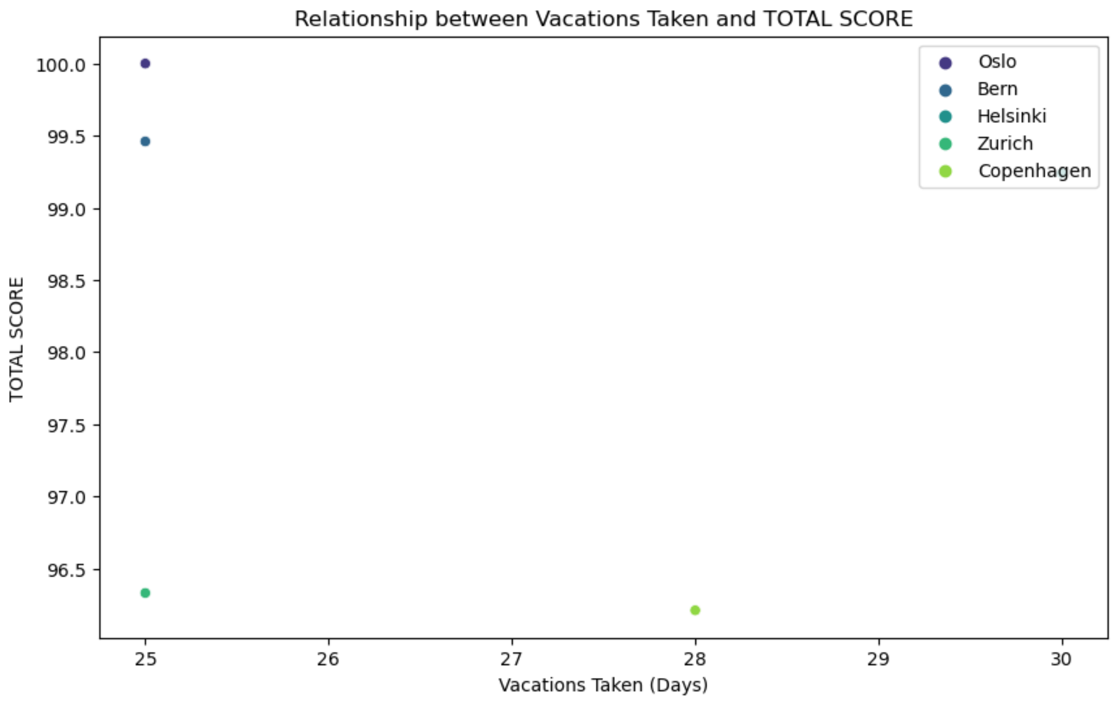
It is no surpise that the highest scoring city, Oslo, has a high minimum vacations days offered, but it comes to my surprise that althoguh Bern and Zurich do offer a minimum of 20 days they still score pretty high on the least becaing second and fourth place respectively.

[ ]:

```

```

[8]:

```
# Visualization 5: Scatter Plot
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='Vacations Taken (Days)', y='TOTAL SCORE',␣
 ↪hue='City', palette='viridis')
plt.xlabel('Vacations Taken (Days)')
plt.ylabel('TOTAL SCORE')
plt.title('Relationship between Vacations Taken and TOTAL SCORE')
plt.legend(loc='upper right')
plt.show()
```



currently and from my perscpective, people value their leisure time highly and there are countries who offer more vacation time than others. In this case we can see that even though Oslo and Bern had some of the least vacation days taken it still score one of the highest in Total Score, meaning that there is more to just vacation days taken that could help a country score higher results.

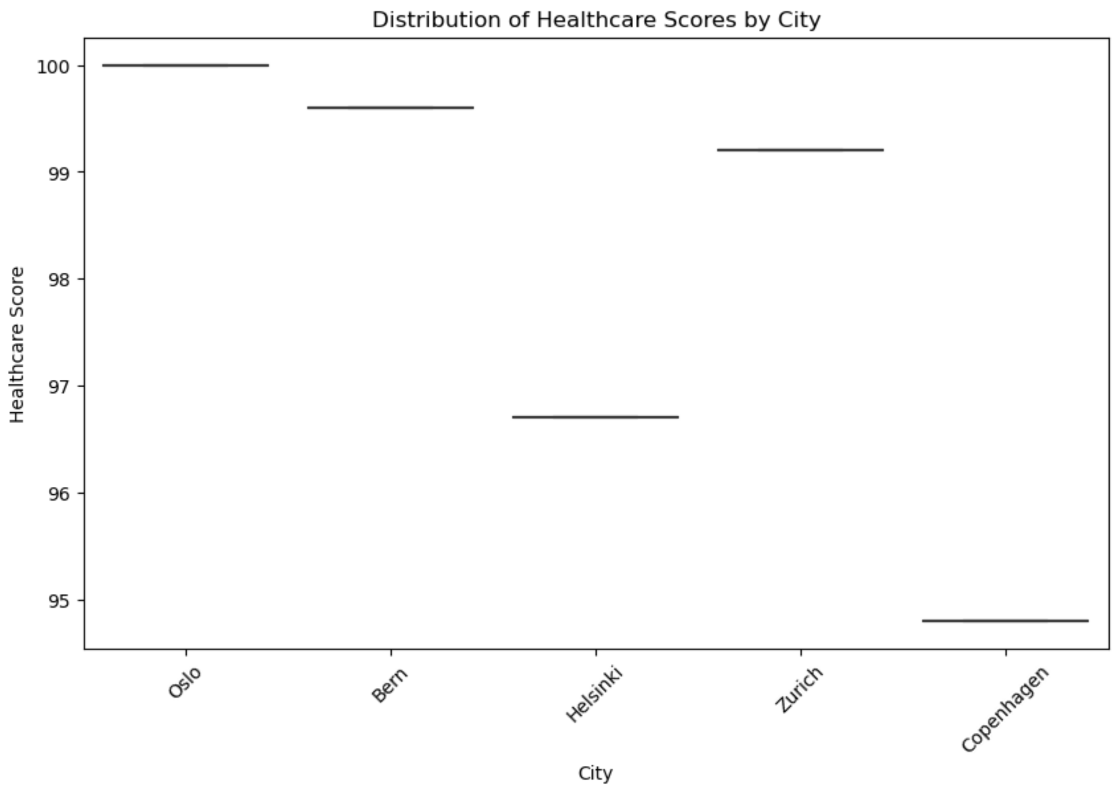
From the previous bar plot we also noticed that Oslo offers a minimum of 25 vacation days, while Bern offers a minimum of 20, but the Bern participants actually took an average of 25 days for vacation taken, which could also lead to the idea that in Bern vacation time can definitely vary or increase if decided.

[ ]:

```
```

[9]:

```
# Visualization 6: Box Plot
plt.figure(figsize=(10, 6))
sns.boxplot(data=df, x='City', y='Healthcare')
plt.xticks(rotation=45)
plt.xlabel('City')
plt.ylabel('Healthcare Score')
plt.title('Distribution of Healthcare Scores by City')
plt.show()
```



Oslo and Bern show that they have a a good practice for leisure time, but now demonstrate that their scores in healthcare are high.

Again demonstrate another reason as to why both cities may have scored as the respective number 1 and number 2 cities to live in.

[ ]:

```

```

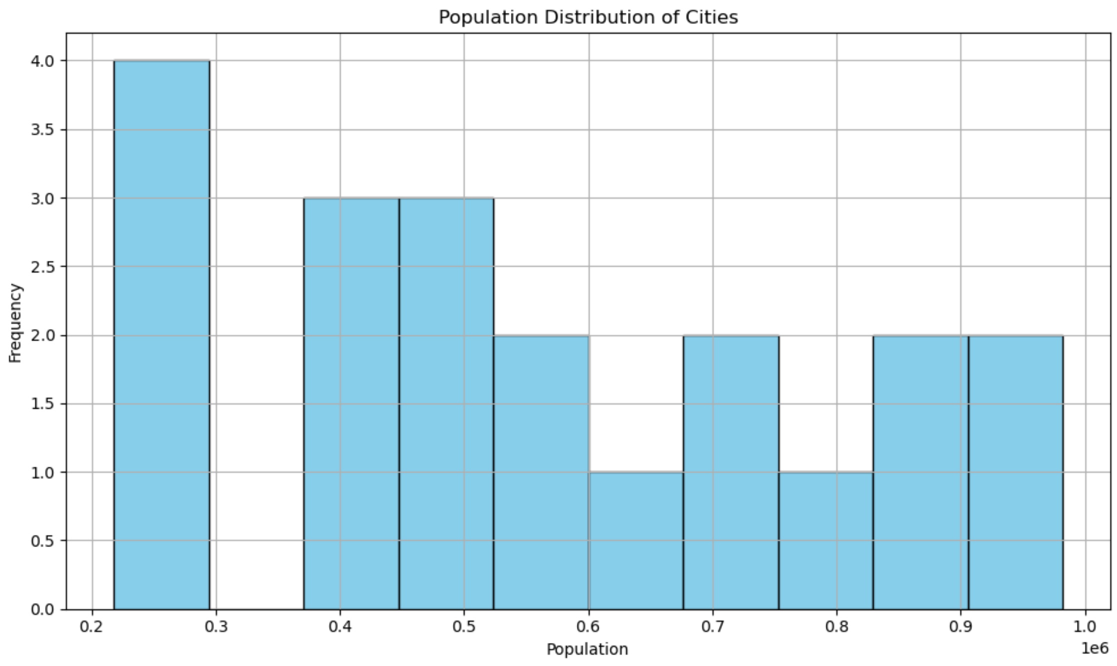
4.1.2 Using random data for more testing:

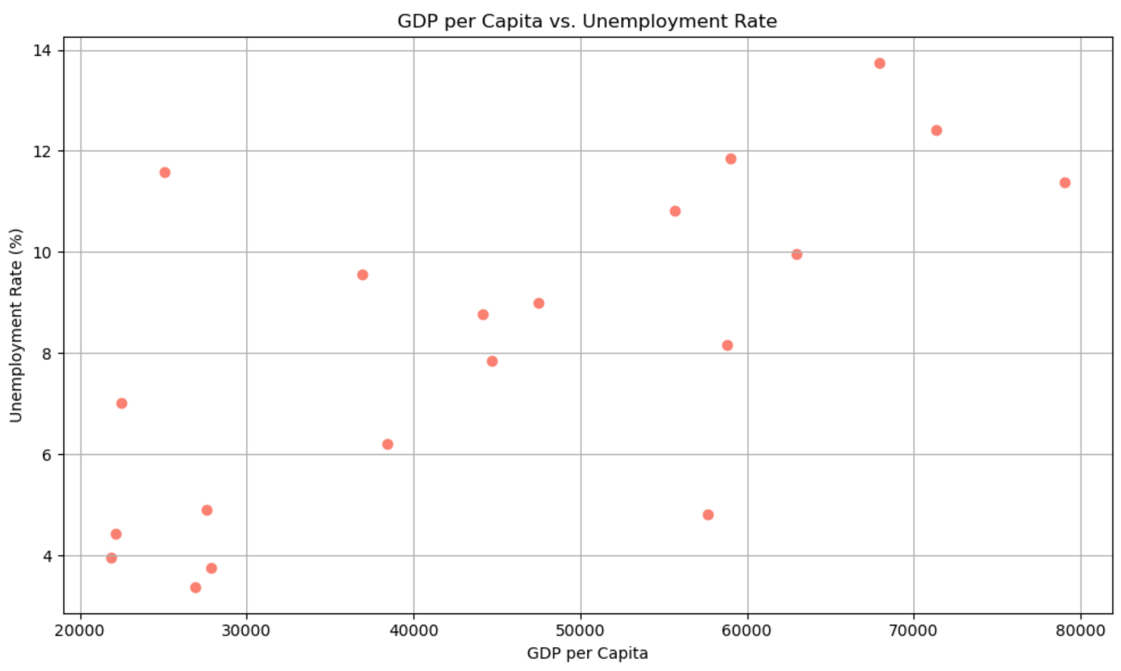
This was done with the aim of testing myself and trying to develop and grow with more practice.

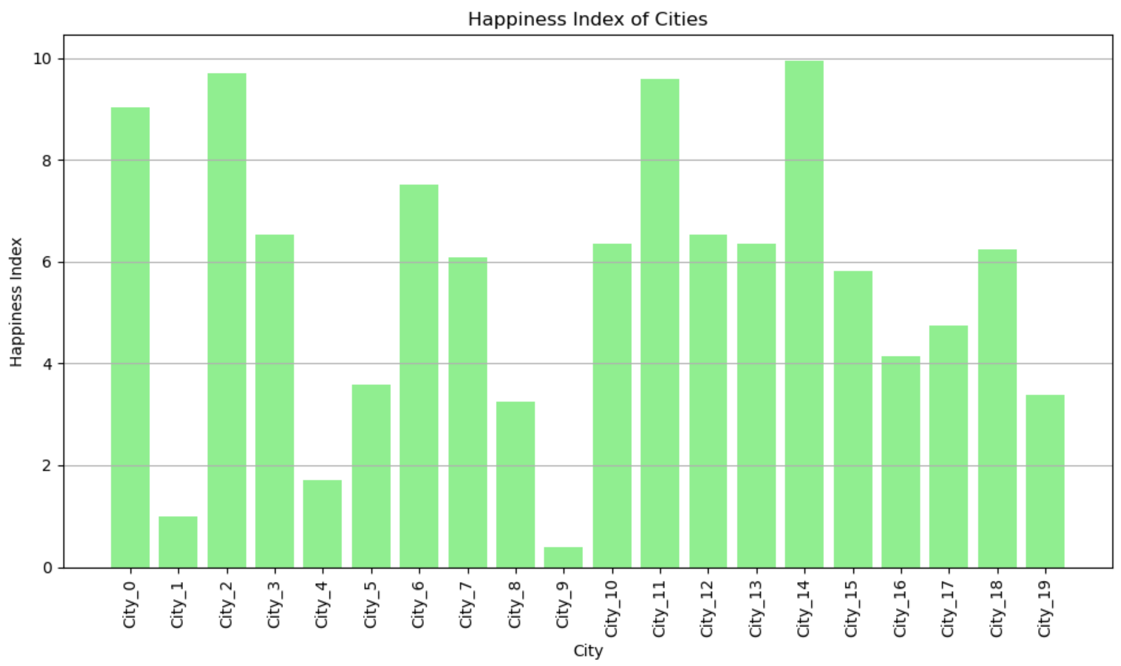
[10]:

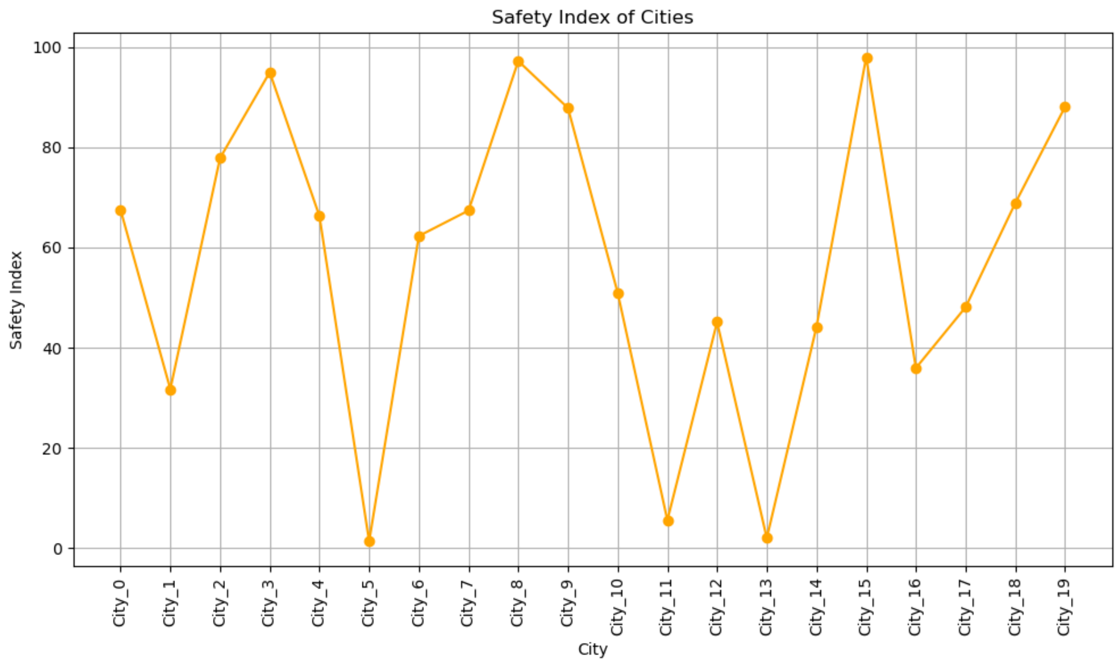
```
import numpy as np

# Example: Generating random data
np.random.seed(0)
num_samples = 20

data = {
    'City': [f'City_{i}' for i in range(num_samples)],
    'Population': np.random.randint(100000, 1000000, size=num_samples),
    'GDP_per_capita': np.random.randint(20000, 80000, size=num_samples),
    'Unemployment_rate': np.random.uniform(2, 15, size=num_samples),
    'Happiness_index': np.random.uniform(0, 10, size=num_samples),
    'Safety_index': np.random.uniform(0, 100, size=num_samples),
}

df = pd.DataFrame(data)

# Visualization 1: Population distribution
plt.figure(figsize=(10, 6))
plt.hist(df['Population'], bins=10, color='skyblue', edgecolor='black')
plt.title('Population Distribution of Cities')
plt.xlabel('Population')
plt.ylabel('Frequency')
plt.grid(True)
plt.tight_layout()
plt.show()

# Visualization 2: GDP per capita vs. Unemployment rate
plt.figure(figsize=(10, 6))
plt.scatter(df['GDP_per_capita'], df['Unemployment_rate'], color='salmon')
plt.title('GDP per Capita vs. Unemployment Rate')
plt.xlabel('GDP per Capita')
plt.ylabel('Unemployment Rate (%)')
plt.grid(True)
plt.tight_layout()
plt.show()

# Visualization 3: Happiness index distribution
plt.figure(figsize=(10, 6))
plt.bar(df['City'], df['Happiness_index'], color='lightgreen')
plt.title('Happiness Index of Cities')
plt.xlabel('City')
plt.ylabel('Happiness Index')
plt.xticks(rotation=90)
plt.grid(axis='y')
plt.tight_layout()
plt.show()

# Visualization 4: Safety index distribution
plt.figure(figsize=(10, 6))
plt.plot(df['City'], df['Safety_index'], marker='o', color='orange',␣
 ↪linestyle='-')
plt.title('Safety Index of Cities')
plt.xlabel('City')
plt.ylabel('Safety Index')
plt.xticks(rotation=90)
plt.grid(True)
plt.tight_layout()
```

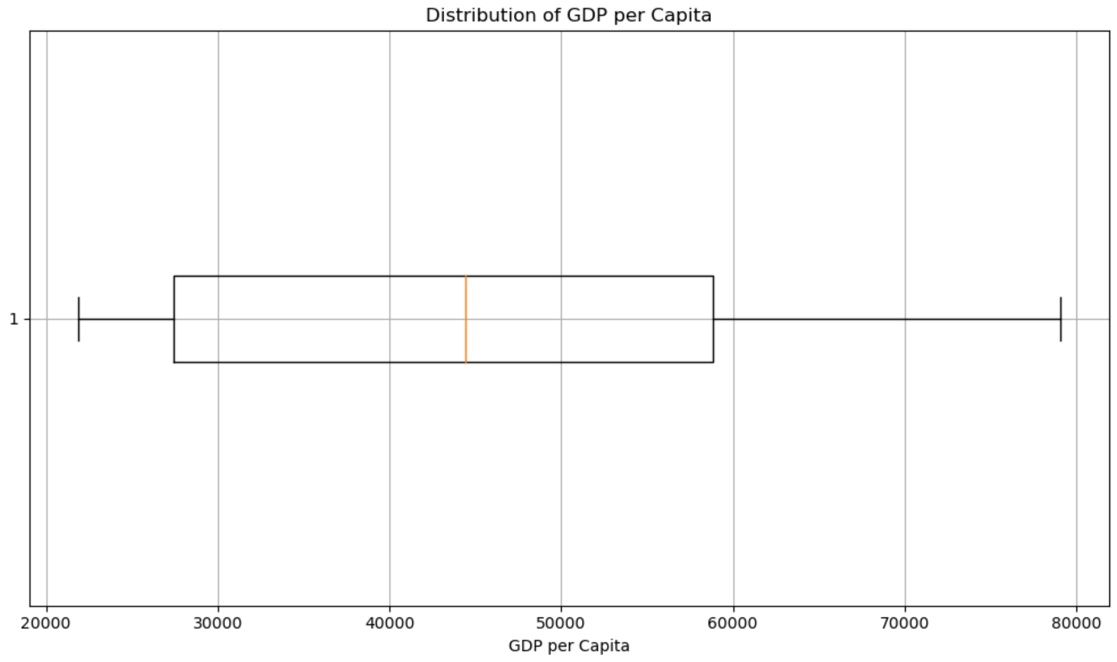
```python
plt.show()

# Visualization 5: GDP per capita distribution
plt.figure(figsize=(10, 6))
plt.boxplot(df['GDP_per_capita'], vert=False)
plt.title('Distribution of GDP per Capita')
plt.xlabel('GDP per Capita')
plt.grid(True)
plt.tight_layout()
plt.show()
```




GDP per Capita vs. Unemployment Rate
Unemployment Rate (%)
GDP per Capita
14
12
10
8
6
4
20000
30000
40000
50000
60000
70000
80000


Happiness Index of Cities
Happiness Index
City
10
8
6
4
2
0
City_0
City_1
City_2
City_3
City_4
City_5
City_6
City_7
City_8
City_9
City_10
City_11
City_12
City_13
City_14
City_15
City_16
City_17
City_18
City_19


Safety Index of Cities
Safety Index
0
20
40
60
80
100
City_0
City_1
City_2
City_3
City_4
City_5
City_6
City_7
City_8
City_9
City_10
City_11
City_12
City_13
City_14
City_15
City_16
City_17
City_18
City_19
City


Distribution of GDP per Capita
1
20000
30000
40000
50000
60000
70000
80000
GDP per Capita

```
[ ]:
```

5 Summary

5.0.1 Issues along the way:

As previously stated my data would not save nor respond as expected, I would run into several issues and would try endlessly to fix and procure a good work, but to no avail.

I have currently given my best to at least be able to turn somehting in rather than skip it or give up altogether.

I will have to pay more attention to instructions and try to follow them as instructed to try to avoid having this issues towards the end when I know that there is a end date.

I noticed that my API and html contained INFORMATION that could be used technically to add more body to my work, but it does not contain "data" as in numbers or calculations. The HTML contains more of a ranking with no data as to why this countries and cities scored the way they did compared to my CSV file that does make calculations explain why some countries or cities are better than others using an average of the data collected.

My project had some hiccups towards the end. As I am unsure if I will be able to finish it all in time I decided to do some testing with randomized data to prove my knowledge and to verify there is not an isue with the actual visualization practices.

5.0.2 Ethical Implications:

As for legal or regulatory guidelines there are none that I am aware of, and none that would come to mind. Another idea I've had is that by knowing what makes a city great I subsequently know what makes a city bad, and this data is for personal use and not for any publications or studies that could lead the misguided results.

All of my data was gathered in an ethical way from credible sources. Some are even used by the government, like the data used previously that is from https://www.getkisi.com/work-life-balance-2022#table Kisi is a cloud-based access control and security platform, one of their main customers is The United States Air Force (USAF), which makes me trust the company more than I would trust any other source.

```
[ ]:
```