AC51011: Big Data Analysis  
**Big Data Processing Assignment**

**Assignment Description:** Given is a dataset (countries.csv) that gives various data about different countries, and 4 problems that require you to perform exploratory data analysis on data in the data set. Write up a report that will contain solutions to all three problems. For each problem, write **how** you have solved it (answering also all the questions that can be found in the description of the problem) and what was the reasoning that you have employed in solving the problem. Make sure to write in the report the Python code that you used to solve the problems (there is no need to attack screenshots this time). Note that if you **only** submit the code without any explanations, you will be severely penalised in terms of the actual grade that you will obtain.  
  
**How You Will Work on the Problems:** This is an **individual** assignment. You are not allowed to collaborate with anyone else on solving these problems, nor to copy the code from elsewhere on the internet. You are allowed to use the lab sheets for solving the problems. If you use help from any other material (e.g. something from the internet), you need to reference it. If your solution is found to be too similar to someone else’s, or to a piece of code from the internet, this might trigger an academic misconduct investigation.   
  
**How Will Your Submission Be Graded:** Grades will be determined by evaluating the overall quality of the report, as well as on the correctness of Python code that you supplied. Note that solving all the problems and answering all the questions specified will give you the grade A5. To get a better grade, you will need to present an especially elegant/good solution, or you will have to do extra work (of your own choice) in addition to the specification.   
  
**What We are Looking for in the Report:** Good explanation of the reasoning that led you to the solution, as well as of any design decisions that you have made.   
  
Problems

1. **Describing Data (10%)**. Consider the following columns in our data set: Population, Population source, Quality of Life Index, Pollution Index. What are the types of each of these columns in terms of   
     
    (i) Numerical discrete, numerical continuous, categorical  
    (ii) Measurement scale (nominal, ordinal, interval, ratio)
2. **Summarising Data (30%).** Consider the same columns as in problem 1.   
     
   (i) What measures of central tendency do you think are the most suitable for each of the columns and why?  
   (ii) Write Python code to compute each of the measures of central tendency from the question (I)  
   (iii) Write Python code to compute the standard deviation of the above four columns for which standard deviation makes sense  
   (iv) Which measure would you use to compare which of the columns, Quality of Life index and Pollution Index, has more variability in the values? Write Python code to calculate this measure and based on that, conclude in which column there is more variability in the values.
3. **Outlier Identification (20%).** Write Python code that will print the names of all the countries that are outliers in term of the density of population per square km3 (Density pop./km2 column).
4. **Correlation (20%).**  Consider the following columns of our dataset: Quality of Life Index, Purchasing Power Index, Pollution Index, Safety Index, Cost of Living Index, Restaurant Price Index and Climate Index.

(i) Which of these columns would you consider (positively or negatively) correlated with Quality of Life Index and would the correlation be strong, normal or weak?  
(ii) Which two columns of the above have the strongest positive correlation between them?

1. **Visualisation (20%).** Write a code to produce graphs in Python to visualise the following information:  
     
   (I) How many countries are in the following bands based on population: small (less than 10,000,000), medium (10,000,000 – 100,000,000) and large (100,000,000+)  
     
   (ii) What percentage of population was obtained using each of the possible population sources (Official quarterly estimate, official provisional figure, Official estimate…)  
     
   (iii) How strong is correlation between Quality of Life Index and Safety Index correlate?

(Iv) How is Cost of Living Index distributed (i.e. how many countries are in each band) in 10 bands of equal length, from minimum to maximum value of the index?

**uAns1)**

First, let’s import the file into Python. I have opted for Jupyter Notebook for running Python code.

import pandas as pd  
data = "C:/Users/Aditya/Downloads/countries.csv"  
df = pd.read\_csv(data)  
df.head()

This will return the first few rows of data. This is done to verify that the file is being read correctly.

A screenshot of a computer screen

Description automatically generated

1. To identify whether the columns we are interested in are numerically discrete, numerically continuous, or categorical, we will create a data frame and insert the names of columns into it. Then, we will use the Pandas Library to determine the result. Here is the code for the same.

df2 = pd.DataFrame()  
df2 = df[["Population", "Population source", "Quality of Life Index", "Pollution Index"]]  
df2.dtypes

This will return us the following output.

Population int64

Population source object

Quality of Life Index float64

Pollution Index float64

dtype: object

This suggests that the column population contains integer data, i.e., numerically discrete data. The population source column returns the data type as an object, which indicates that it contains categorical data. The quality-of-life index and the pollution index column return float64 data type, meaning that it includes numerically continuous data.

1. Next, we talk about the measurement scale.

**Nominal**

It is the first and the most superficial level of scale of measurement. When classifying and labelling variables, they are categorised qualitatively. This involves dividing them into named groups without any quantitative significance. It is important to note that when different categories are labelled using numbers, these numbers do not hold any numerical value.

The above code shows that only the population source column returned the data type as a categorical data object. But it does not have any order or ranking to it. Hence, it is a nominal variable.

**Ordinal**

Ordinal data can be put into categories with a natural rank order. However, the distances between the categories are not necessarily equal. They have a hierarchy to them. Examples of ordinal data can be:

1. High
2. Medium
3. Low

None of the above four columns have data that can be ranked. So, there is no ordinal data in the selected columns of the data set.

**Interval Data**

Data measured along a numerical scale with equal distances between adjacent values is called interval data. These distances are known as 'intervals'. An interval scale differs from a ratio scale as it does not have a true zero point. On an interval scale, zero is a relative point rather than a complete absence of the variable. Also, mathematical operations can be performed on this data.

Out of 4 columns, we have only 3 columns with quantitative data. They are population, quality of life index, and pollution index. All these columns have a true zero. The population being zero in a country means there are no people there. Next, there is no quality of life without people, meaning the population is absent. Hence, it does have an absolute or true zero. The same goes for the pollution index.

Therefore, all of them are ratio data.

**Ratio Data**

Like interval data, they also have meaningful mathematical operations that can be performed on them, but additionally, they also have a true zero. For example, the Kelvin scale for measuring temperature has an absolute zero on its scale. That is, nothing can be colder than 0 K.

As we discussed earlier, the four columns' population, quality of life index and pollution index are ratio data.

**Ans2)**

1. The population source column comprises categorical data. Therefore, only mode can be calculated for it. It has no numerical/quantitative data, so the mean and median cannot be calculated.

Population data is quantitative; hence, all the measures of central tendency can be applied. The same goes for the quality-of-life index and pollution index.

The mean is considered the best measure of central tendency when the data is normally distributed, and the median is more useful when the data is skewed or has outliers. Therefore, we will first determine if the population, quality of life index, and pollution index are normally distributed.

We will use the Shapiro test to check the same for all the three columns.

Code:

1. Population

from scipy.stats import shapiro  
stat, p\_value = shapiro(df['Population’])  
# Check the p-value  
alpha = 0.05  
print(f'Shapiro-Wilk Test:\nStatistic={stat}, p-value={p\_value}')  
if p\_value > alpha:  
 print("The data appears to be normally distributed.")  
else:  
 print("The data does not appear to be normally distributed.")

Output:

A black text on a white background

Description automatically generated

Therefore, population data is not normally distributed. Hence, the best central tendency to measure would be the median.

1. Quality of Life Index

from scipy.stats import shapiro  
stat, p\_value = shapiro(df[' Quality of Life Index'])  
# Check the p-value  
alpha = 0.05  
print(f'Shapiro-Wilk Test:\nStatistic={stat}, p-value={p\_value}')  
if p\_value > alpha:  
 print("The data appears to be normally distributed.")  
else:  
 print("The data does not appear to be normally distributed.")

Output:

A black text on a white background

Description automatically generated

The Quality of Life Index data is found to be normally distributed using the Shapiro-Wilk test. Hence, the mean is the best measure of central tendency that can be used here.

1. Pollution Index

from scipy.stats import shapiro  
stat, p\_value = shapiro(df['Pollution Index'])  
  
# Check the p-value  
alpha = 0.05  
print(f'Shapiro-Wilk Test:\nStatistic={stat}, p-value={p\_value}')  
if p\_value > alpha:  
 print("The data appears to be normally distributed.")  
else:  
 print("The data does not appear to be normally distributed.")

Output:

A black text on a white background

Description automatically generated

Again, the pollution index data is found to be not normally distributed. Hence, the median is the best measure of central tendency to be used here.

1. Now, we will determine the mode for population source, mean and median for Population, Quality of Life Index, and Pollution Index.
2. Population Source

First, we import mode from the Python statistics package to measure the mode. Then, we put the population source in a new data frame and then use the mode() function to determine the mode.

Code:

from statistics import mode  
df3 = pd.DataFrame()  
df3 = df[["Population source"]]  
mode\_pop\_source = df3.mode()  
print(mode\_pop\_source)

Output:

A screenshot of a computer

Description automatically generated

The output shows that official estimate has been a popular source for collecting population data.

1. Population

We will now use Python code to determine the mean and median of population.

Code:

*# Creating a new data frame and adding population column to it*

df4 = pd.DataFrame()  
df4 = df[["Population"]]  
mean\_population = df4.mean()  
median\_population = df4.median()  
print(mean\_population)  
print(median\_population)

Output:

A black numbers and numbers on a white background

Description automatically generated

The average or mean population was found to be 8.082307e+07, and the median population was found to be 18019642.0.

As we found the population data skewed earlier, we can see that mean>median.

The data type is changed to float 64 from the integer because of calculations.

1. We will use the same code to determine the mean and median Quality of Life and Pollution Index.

Code:

df5 = pd.DataFrame()  
df5 = df[["Quality of Life Index"]]  
mean\_quality = df5.mean()  
median\_quality = df5.median()  
print(mean\_quality)  
print(median\_quality)

Output:

A close up of text

Description automatically generated

Here, we can see that the mean and median are almost similar. We found the quality of life index to be normally distributed, so we can confirm this after observing the result.

1. Pollution Index

Code:

df6 = pd.DataFrame()  
df6 = df[["Pollution Index"]]  
mean\_pollution = df6.mean()  
median\_pollution = df6.median()  
print(mean\_pollution)  
print(median\_pollution)

Output:

A number of numbers on a white background

Description automatically generated

As we found the pollution index data skewed earlier, we can see that mean<median.

1. Standard deviation applies to only population, quality of life index and pollution index and not the population source because it is a categorical data.

Below is the code for calculating standard deviation for all the necessary columns.

Code:

std\_population = df4. std()  
std\_quality = df5.std()  
std\_pollution = df6.std()  
print(f"The standard deviation of population is: {std\_population}")  
print(f"The standard deviation of quality of life index is: {std\_quality}")  
print(f"The standard deviation of pollution is: {std\_pollution}")

Output:

A white background with black text

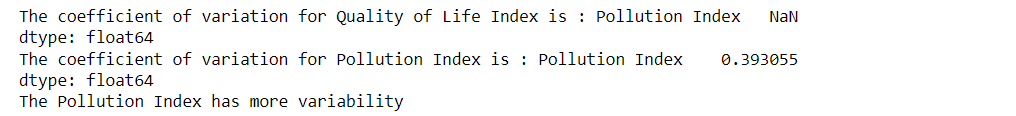
Description automatically generated

1. We will use the coefficient of variation to compare the variability of the Quality of Life and Pollution Index. The ratio of Standard deviation to the mean gives it.

Code:

#Coefficient of variation is the ratio of standard deviation to mean

cv\_quality\_of\_life\_index = std\_quality/mean\_quality  
cv\_pollution\_index = std\_pollution/mean\_pollution  
cv\_quality\_of\_life\_index = cv\_quality\_of\_life\_index.reindex(index=cv\_pollution\_index.index)  
  
print(f'The coefficient of variation for Quality of Life Index is : {cv\_quality\_of\_life\_index}')  
print(f'The coefficient of variation for Pollution Index is : {cv\_pollution\_index}')  
  
#Comparing variability  
if cv\_quality\_of\_life\_index.any() > cv\_pollution\_index.any():  
 print("The Quality of Life Index has more variability")   
else:  
 print("The Pollution Index has more variability")Output:



In the above code, it was necessary to reindex the two coefficients to be comparable. Also, using the .any() function was necessary because the pandas would consider the truth value of the series ambiguous otherwise.

**Ans3)**

An outlier is an observation that significantly deviates from a dataset's overall pattern or distribution, typically defined as a data point that falls outside a predetermined range, often based on statistical measures like the interquartile range (IQR) or standard deviation.

Code:

df7 = df[['Country', 'Density pop./km2']]

def outlier\_detection(df7):  
 q1 = np.percentile(df7['Density pop./km2'], 25)  
 q3 = np.percentile(df7['Density pop./km2'], 75)  
 # Calculating interquartile range  
 IQR = q3 - q1  
 # Calculating lower and upper limit  
 lwr\_limit = q1 - 1.5 \* IQR  
 upr\_limit = q3 + 1.5 \* IQR  
 # Defining outliers  
 outliers = (df7['Density pop./km2'] > upr\_limit) | (df7['Density pop./km2'] < lwr\_limit)  
 # Get countries with outlier values  
 outlier\_countries = df7.loc[outliers, 'Country'].tolist()  
 return outlier\_countries

outlier\_countries = outlier\_detection(df7)

Output:

A screenshot of a computer

Description automatically generated

Using the above code, we create a function for detecting outliers in population density. The function utilises interquartile range to determine the outliers. The interquartile range (IQR) is a measure of statistical dispersion, defined as the difference between the third quartile (Q3) and the first quartile (Q1), representing the middle 50% of the data in a dataset.

Then we use the .loc[] function to collect only the countries where the outliers Boolean is true and add them to a python list. Finally, print(outlier\_countries) return the list of countries with outlier population density.

**Ans4)**

Code:

# Creating Dataframe  
df8 = pd.DataFrame()  
df8 = df[['Quality of Life Index', 'Purchasing Power Index', 'Pollution Index', 'Safety Index', 'Cost of Living Index', 'Restaurant Price Index', 'Climate Index']]  
# Calculating correlation with Quality of Life Index  
corr\_quality\_index = df8.corrwith(df8['Quality of Life Index'], method = 'pearson')  
df9 = pd.DataFrame(columns = ['Variable', 'Correlation', 'Strength'])  
  
# Categorize correlation strength  
for column, correlation in corr\_quality\_index.items():  
 if correlation == 1:  
 strength = "Perfectly Positive"  
 elif correlation > 0.7:  
 strength = "strong positive"  
 elif 0.7 >=correlation > 0.3:  
 strength = "normal positive"  
 elif correlation < -0.7:  
 strength = "strong negative"  
 elif correlation < -0.3:  
 strength = "normal negative"  
 else:  
 strength = "weak or no"  
 df9 = df9.append({'Variable': column, 'Correlation': correlation, 'Strength': strength}, ignore\_index=True)  
print(df9)

Output:

A white background with black text

Description automatically generated

The above code creates a new data frame of interested columns and calculates the correlation with the “Quality of Life Index Column”. Furthermore, it also categorises the variables based on the

Except for the pollution index, all the other variables were positively correlated with the Quality-of-Life Index.

Purchasing Power Index, Restaurant Price Index and Cost of Living index were strongly correlated with quality-of-life Index.

The Safety Index was found to be normally correlated with the quality-of-life index.

The Climate Index was weakly correlated with the quality-of-life index, meaning it does not affect it.

To determine the variables with the highest positive correlation, we wrote a code that calculates correlation among all and plots the correlation heatmap to verify that.

Code:

import numpy as np

corr\_matrix = df8.corr().abs()  
  
# Exclude the diagonal by setting diagonal elements to zero  
np.fill\_diagonal(corr\_matrix.values, 0)  
  
# Find the indices of the maximum correlation value  
max\_corr\_index = np.unravel\_index(np.argmax(corr\_matrix.values), corr\_matrix.shape)  
  
# Extract the column names and the maximum correlation value  
column1, column2, max\_corr\_value = df8.columns[max\_corr\_index[0]], df8.columns[max\_corr\_index[1]], corr\_matrix.iloc[max\_corr\_index]  
  
# Display the result  
print(f"The maximum correlation (excluding diagonal) is between columns {column1} and {column2}: {max\_corr\_value}")

**Output:**

****

Code for heatmap

**import** seaborn **as** sns  
**import** matplotlib.pyplot **as** plt  
  
*# Assuming corr\_matrix is your correlation matrix*  
mask = np.triu(np.ones\_like(corr\_matrix, dtype=bool))  
  
sns.set(style="white")  
plt.figure(figsize=(12, 8))  
sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm', linewidths=0, cbar\_kws={'label': 'Correlation Coefficient'},   
 annot\_kws={'size': 15}, mask=mask)  
plt.title('Correlation Heatmap', fontsize=20, pad=20)  
plt.xlabel('Index', fontsize=16)  
plt.ylabel('Index', fontsize=16)  
sns.despine(left=True, bottom=True)  
plt.show()

A graph of heatmap

Description automatically generated

As we can verify from the above correlation heatmap, the highest positive correlation is between the Cost-of-Living Index and the Restaurant Price Index.

**Ans5)**

# Adjust the bin   
bins = [0, 10000000, 100000000, float('inf')]  
labels = ['Small (<10M)', 'Medium (10M-100M)', 'Large (100M+)']  
  
df['Population Band'] = pd.cut(df['Population'], bins=bins, labels=labels, right=False)  
  
plt.figure(figsize=(10, 6))  
sns.countplot(x='Population Band', data=df, palette='viridis')  
plt.title('Number of Countries in Population Bands', fontsize = 20)  
plt.xlabel('Population bands', fontsize=16)  
plt.ylabel('Count', fontsize=16)  
plt.xticks(fontsize=14)  
plt.yticks(fontsize=14)  
sns.despine(left=True, bottom=True)  
plt.show()

A graph of different colored squares

Description automatically generated

We can observe that the number of countries with a population of less than 10 million is 30, classified as a small population. Around thirty-five countries fall under the medium population bands ranging from 10 million to 100 million. Finally, around thirteen countries fall under the large population bands greater than 100 million.



We will use two plots to get the percentage of the population obtained in the dataset by source. Firstly, the bar plot of percentages by population source in descending order to understand which source was used the greatest number of times.

Next, we plot pie chart of the population data source to understand the composition of different sources.

# DataFrame with columns 'Country', 'Population', 'Population source'  
sns.set(style='whitegrid')  
plt.figure(figsize=(16, 8))  
sns.barplot(y='Population source', x='Population', data=df,   
 estimator=lambda x: sum(x) / df['Population'].sum() \* 100,   
 palette='BuGn\_r', order=df.groupby('Population source')['Population'].sum().sort\_values(ascending=False).index, ci = None)  
plt.title('Percentage of Population by Source', fontsize=20)  
plt.xlabel('Percentage of Total Population', fontsize=16)  
plt.ylabel('Population source', fontsize=16)  
plt.xticks(fontsize=14)  
plt.yticks(fontsize=14)  
sns.despine(left=True, bottom=True)  
plt.show()

A graph of a number of people

Description automatically generated

Code2: Pie Chart

sns.set(style='whitegrid')  
plt.figure(figsize=(12, 12))  
  
*# Calculate the percentage of population by source*  
population\_percentages = df.groupby('Population source')['Population'].sum() / df['Population'].sum() \* 100  
  
*# Group categories with less than 1% into 'Others'*  
threshold = 2  
population\_percentages\_others = population\_percentages[population\_percentages >= threshold]  
population\_percentages\_others['Others'] = population\_percentages[population\_percentages < threshold].sum()  
  
*# Plotting as a pie chart*  
plt.pie(population\_percentages\_others, labels=None, autopct='%1.1f%%', startangle=140, colors=sns.color\_palette('GnBu'))  
  
*# Add a legend*  
plt.legend(population\_percentages\_others.index, title='Population source', loc='center left', bbox\_to\_anchor=(1, 0.5), fontsize='large')  
  
*# Add a title*  
plt.title('Percentage of Population by Source', fontsize=20)  
  
*# Show the pie chart*  
plt.show()

A blue pie chart with green and blue numbers

Description automatically generated

To make the pie chart less cluttered, we have created a category ‘others’ that consists of all the population sources that contribute less than 2% of data and are represented on the pie chart as a sum.

To plot the correlation between the Quality-of-Life Index and the Safety Index, we will first plot a scatter plot to understand the general direction of the correlation, positive or negative, and then we will determine the strength using the correlation heatmap.

Code1: Scatterplot

plt.figure(figsize=(8, 8))  
  
# Scatter plot with trendline  
sns.regplot(y='Quality of Life Index', x='Safety Index', data=df, scatter\_kws={'color': 'red'}, line\_kws={'color': 'blue'})  
  
# Add a title  
plt.title('Scatter Plot with Trendline between Quality of Life Index and Safety Index', fontsize=20)  
  
# Customize axis labels and ticks  
plt.xlabel('Quality of Life Index', fontsize=14)  
plt.ylabel('Safety Index', fontsize=14)  
plt.xticks(fontsize=12)  
plt.yticks(fontsize=12)  
  
# Remove grids and spines for aesthetics  
plt.grid(False)  
sns.despine(left=True, bottom=True)  
  
# Show the scatter plot  
plt.show()

**Output:**

A diagram of a graph

Description automatically generated

Based on the scatter plot above, we can conclude a positive correlation between the safety index and the quality-of-life index. As the safety index increases, the quality-of-life index also increases. This aligns with the general observation that countries with higher safety levels have better quality of life.

Next, we plot a correlation heatmap to assign a quantitative value to the correlation strength among the two indices.

Code2: Correlation Heatmap

plt.figure(figsize=(8, 8))  
sns.heatmap(df[['Quality of Life Index', 'Safety Index']].corr(), annot=True, cmap='Reds', linewidths=0, cbar\_kws={'label': 'Correlation Coefficient'}, annot\_kws={'size': 18})  
plt.title('Correlation Heatmap between Quality of Life Index and Safety Index', fontsize=20)  
plt.xticks(fontsize=14)  
plt.yticks(fontsize=14)  
sns.despine(left=True, bottom=True)  
plt.show()

A diagram of a heatmap

Description automatically generated

After plotting the correlation heatmap between the safety index and the quality-of-life index, it conforms with the previous scatter plot with the coefficient of correlation being 0.59. As discussed in question 4, part 1, the correlation between the two indexes usually is positive.

We first calculated its range to see the distribution of the cost-of-living index in 10 bands of equal length. Then, the range was split into ten bins of similar size, and the count plot was used to see the number of countries in each bin.

# Assuming df is your DataFrame  
max\_coi = df['Cost of Living Index'].max()  
min\_coi = df['Cost of Living Index'].min()  
range\_coi = max\_coi - min\_coi  
  
# Calculate bandwidth  
band\_width = range\_coi / 10  
  
# Create bins  
bins = [min\_coi + i \* band\_width for i in range(11)]  
  
# Bin the data  
df['COI\_Band'] = pd.cut(df['Cost of Living Index'], bins=bins, labels=[f"Band {i+1}" for i in range(10)], include\_lowest=True)  
  
# Count the number of countries in each band and sort  
band\_counts = df['COI\_Band'].value\_counts().sort\_index()  
sorted\_band\_counts = band\_counts.sort\_values()  
  
# Create a dictionary for the legend  
legend\_labels = {f"{band} ({count} countries)": count for band, count in sorted\_band\_counts.items()}  
  
# Print the band ranges

**for** band, range\_values **in** band\_ranges.items():

print(band, ":", range\_values)

# Plotting  
sns.set(style = 'darkgrid')  
plt.figure(figsize=(10, 6)) # Set the size of the plot  
ax = sorted\_band\_counts.plot(kind='bar')  
plt.title('Number of Countries per Cost of Living Index Band')  
plt.xlabel('Cost of Living Index Band')  
plt.ylabel('Number of Countries')  
  
# Add legend outside the plot  
ax.legend([plt.Rectangle((0,0),1,1,fc="gray") for label in legend\_labels.keys()],  
 [label for label in legend\_labels.keys()], bbox\_to\_anchor=(1.05, 1), loc='upper left')  
  
plt.tight\_layout()  
plt.show()

**Output:**

A table of numbers with black numbers

Description automatically generated

A graph of a number of countries/regions

Description automatically generated

After observing the count plot, we can say that most countries lie in band 2 of the cost-of-living index with a value of 23. As we calculated the ranges of different bands in the code, we can say that most countries have the cost of living index in Band 2 i.e. (31.61, 35.31).