In [1]:

```
#Load the data into a DataFrame and print the results
#Query the number of rows
#Print the column headers
#Print the data types
#Print the index
#WHO_first9cols.CSV


#load pd
import pandas as pd

#load csv
#WHO_first9cols.CSV
df = pd.read_csv('WHO_first9cols.CSV')

#number of rows
num_rows = len(df)

#column headers
column_headers = df.columns.tolist()

#data types
data_types = df.dtypes

#index
index_info = df.index
#print number of rows, column headers, data types, and index
print(f"Number of Rows: {num_rows}")
print(f"Column Headers: {column_headers}")
print(f"Data Types:\n{data_types}")
print(f"Index Information: {index_info}")
```

```
Number of Rows: 202
Column Headers: ['Country', 'CountryID', 'Continent', 'Adolescent fertility rate
(%)', 'Adult literacy rate (%)', 'Gross national income per capita (PPP international
$)', 'Net primary school enrolment ratio female (%)', 'Net primary school enrolment r
atio male (%)', 'Population (in thousands) total']
Data Types:
Country                                                 object
CountryID                                                int64
Continent                                                int64
Adolescent fertility rate (%)                          float64
Adult literacy rate (%)                                float64
Gross national income per capita (PPP international $)    float64
Net primary school enrolment ratio female (%)          float64
Net primary school enrolment ratio male (%)            float64
Population (in thousands) total                        float64
dtype: object
Index Information: RangeIndex(start=0, stop=202, step=1)
```

In [2]:

```
#Using the same file, select the “Country” column and return its data type along with

#load pd
import pandas as pd

#load csv in df
#WHO_first9cols.CSV
df = pd.read_csv('WHO_first9cols.CSV')
```

```
#only Country column
country_column = df['Country']

#data type
data_type = country_column.dtype

#series shape
shape = country_column.shape

#index
index_info = country_column.index

#values
values = country_column.values

#name
name = country_column.name

#print data type, series shape, index, values and name
print(f"Data Type: {data_type}")
print(f"Series Shape: {shape}")
print(f"Index Information: {index_info}")
print(f"Values: {values}")
print(f"Name: {name}")
```

```
Data Type: object
Series Shape: (202,)
Index Information: RangeIndex(start=0, stop=202, step=1)
Values: ['Afghanistan' 'Albania' 'Algeria' 'Andorra' 'Angola'
 'Antigua and Barbuda' 'Argentina' 'Armenia' 'Australia' 'Austria'
 'Azerbaijan' 'Bahamas' 'Bahrain' 'Bangladesh' 'Barbados' 'Belarus'
 'Belgium' 'Belize' 'Benin' 'Bermuda' 'Bhutan' 'Bolivia'
 'Bosnia and Herzegovina' 'Botswana' 'Brazil' 'Brunei Darussalam'
 'Bulgaria' 'Burkina Faso' 'Burundi' 'Cambodia' 'Cameroon' 'Canada'
 'Cape Verde' 'Central African Republic' 'Chad' 'Chile' 'China' 'Colombia'
 'Comoros' 'Congo, Dem. Rep.' 'Congo, Rep.' 'Cook Islands' 'Costa Rica'
 "Cote d'Ivoire" 'Croatia' 'Cuba' 'Cyprus' 'Czech Republic' 'Denmark'
 'Djibouti' 'Dominica' 'Dominican Republic' 'Ecuador' 'Egypt'
 'El Salvador' 'Equatorial Guinea' 'Eritrea' 'Estonia' 'Ethiopia' 'Fiji'
 'Finland' 'France' 'French Polynesia' 'Gabon' 'Gambia' 'Georgia'
 'Germany' 'Ghana' 'Greece' 'Grenada' 'Guatemala' 'Guinea' 'Guinea-Bissau'
 'Guyana' 'Haiti' 'Honduras' 'Hong Kong, China' 'Hungary' 'Iceland'
 'India' 'Indonesia' 'Iran (Islamic Republic of)' 'Iraq' 'Ireland'
 'Israel' 'Italy' 'Jamaica' 'Japan' 'Jordan' 'Kazakhstan' 'Kenya'
 'Kiribati' 'Korea, Dem. Rep.' 'Korea, Rep.' 'Kuwait' 'Kyrgyzstan'
 "Lao People's Democratic Republic" 'Latvia' 'Lebanon' 'Lesotho' 'Liberia'
 'Libyan Arab Jamahiriya' 'Lithuania' 'Luxembourg' 'Macao, China'
 'Macedonia' 'Madagascar' 'Malawi' 'Malaysia' 'Maldives' 'Mali' 'Malta'
 'Marshall Islands' 'Mauritania' 'Mauritius' 'Mexico'
 'Micronesia (Federated States of)' 'Moldova' 'Monaco' 'Mongolia'
 'Montenegro' 'Morocco' 'Mozambique' 'Myanmar' 'Namibia' 'Nauru' 'Nepal'
 'Netherlands' 'Netherlands Antilles' 'New Caledonia' 'New Zealand'
 'Nicaragua' 'Niger' 'Nigeria' 'Niue' 'Norway' 'Oman' 'Pakistan' 'Palau'
 'Panama' 'Papua New Guinea' 'Paraguay' 'Peru' 'Philippines' 'Poland'
 'Portugal' 'Puerto Rico' 'Qatar' 'Romania' 'Russia' 'Rwanda'
 'Saint Kitts and Nevis' 'Saint Lucia' 'Saint Vincent and the Grenadines'
 'Samoa' 'San Marino' 'Sao Tome and Principe' 'Saudi Arabia' 'Senegal'
 'Serbia' 'Seychelles' 'Sierra Leone' 'Singapore' 'Slovakia' 'Slovenia'
 'Solomon Islands' 'Somalia' 'South Africa' 'Spain' 'Sri Lanka' 'Sudan'
 'Suriname' 'Swaziland' 'Sweden' 'Switzerland' 'Syria' 'Taiwan'
 'Tajikistan' 'Tanzania' 'Thailand' 'Timor-Leste' 'Togo' 'Tonga'
 'Trinidad and Tobago' 'Tunisia' 'Turkey' 'Turkmenistan' 'Tuvalu' 'Uganda'
 'Ukraine' 'United Arab Emirates' 'United Kingdom'
 'United States of America' 'Uruguay' 'Uzbekistan' 'Vanuatu' 'Venezuela'
 'Vietnam' 'West Bank and Gaza' 'Yemen' 'Zambia' 'Zimbabwe']
Name: Country
```

In [11]:

```
print(data.columns)
```

```
Index(['Yearly Mean Total Sunspot Number', 'Yearly Mean Standard Deviation',
       'Number of Observations', 'Definitive/Provisional Indicator'],
      dtype='object')
```

In [23]:

```
#Using the Quandl API, import the data
#Print the head() and tail()
#Query for the last value using the last date
#Query the date with date strings in the YYYYMMDD format
#Query with a Boolean, where the number of observations is greater than the mean numbe
#Query with a Boolean, where the number of sunspots is greater than the mean number of

#for the quandl api, need to load quandl library
#!pip install quandl #done

#load quand1, and pd
import quandl
import pandas as pd

#api key kgECEhP4wEwUFD7MW_Ae
quandl.ApiConfig.api_key = "kgECEhP4wEwUFD7MW_Ae"


#Index(['Yearly Mean Total Sunspot Number', 'Yearly Mean Standard Deviation', 'Number
#sunspot
dataset_code = "SIDC/SUNSPOTS_A"
data = quandl.get(dataset_code)

#head and tail
print("head:")
print(data.head())
print("\nTail:")
print(data.tail())

#query for last value w/ last date
#Yearly Mean Total Sunspot Number
last_date = data.index[-1]
last_value = data.loc[last_date]['Yearly Mean Total Sunspot Number']
print(f"\nLast date: {last_date}")
print(f"last value: {last_value}")

#date strings with yyyymmdd format
date_string = "20230918"
date_query = data[data.index.strftime('%Y%m%d') == date_string]
print(f"\nDate query for {date_string}:")
print(date_query)

#boolean, with # of observations is > mean
mean_observations = data['Number of Observations'].mean()

#query # of observations is > mean
observations_greater_than_mean = data[data['Number of Observations'] > mean_observatic
print("\nQuery with Number of Observations > Mean:")
print(observations_greater_than_mean)

#sunspots is > mean
sunspots_greater_than_mean = data[data['Yearly Mean Total Sunspot Number'] > data['Yea
print("\nQuery with Sunspots > Mean:")
print(sunspots_greater_than_mean)
```

```
head:
            Yearly Mean Total Sunspot Number  Yearly Mean Standard Deviation  \
Date
1700-12-31                               8.3                             NaN
1701-12-31                              18.3                             NaN
1702-12-31                              26.7                             NaN
1703-12-31                              38.3                             NaN
1704-12-31                              60.0                             NaN

            Number of Observations  Definitive/Provisional Indicator
Date
1700-12-31                     NaN                               1.0
1701-12-31                     NaN                               1.0
1702-12-31                     NaN                               1.0
1703-12-31                     NaN                               1.0
1704-12-31                     NaN                               1.0

Tail:
            Yearly Mean Total Sunspot Number  Yearly Mean Standard Deviation  \
Date
2016-12-31                              39.8                             3.9
2017-12-31                              21.7                             2.5
2018-12-31                               7.0                             1.1
2019-12-31                               3.6                             0.5
2020-12-31                               8.8                             4.1

            Number of Observations  Definitive/Provisional Indicator
Date
2016-12-31                  9940.0                               1.0
2017-12-31                 11444.0                               1.0
2018-12-31                 12611.0                               1.0
2019-12-31                 12884.0                               1.0
2020-12-31                 14440.0                               1.0

Last date: 2020-12-31 00:00:00
last value: 8.8

Date query for 20230918:
Empty DataFrame
Columns: [Yearly Mean Total Sunspot Number, Yearly Mean Standard Deviation, Number of
Observations, Definitive/Provisional Indicator]
Index: []

Query with Number of Observations > Mean:
            Yearly Mean Total Sunspot Number  Yearly Mean Standard Deviation  \
Date
1981-12-31                             198.9                            13.1
1982-12-31                             162.4                            12.1
1983-12-31                              91.0                             7.6
1984-12-31                              60.5                             5.9
1985-12-31                              20.6                             3.7
1986-12-31                              14.8                             3.5
1987-12-31                              33.9                             3.7
1988-12-31                             123.0                             8.4
1989-12-31                             211.1                            12.8
1990-12-31                             191.8                            11.2
1991-12-31                             203.3                            12.7
1992-12-31                             133.0                             8.9
1993-12-31                              76.1                             5.8
1994-12-31                              44.9                             4.4
```

```
1995-12-31                              25.1                             3.7
1996-12-31                              11.6                             3.1
1997-12-31                              28.9                             3.6
1998-12-31                              88.3                             6.6
1999-12-31                             136.3                             9.3
2000-12-31                             173.9                            10.1
2001-12-31                             170.4                            10.5
2002-12-31                             163.6                             9.8
2003-12-31                              99.3                             7.1
2004-12-31                              65.3                             5.9
2005-12-31                              45.8                             4.7
2006-12-31                              24.7                             3.5
2007-12-31                              12.6                             2.7
2008-12-31                               4.2                             2.5
2009-12-31                               4.8                             2.5
2010-12-31                              24.9                             3.4
2011-12-31                              80.8                             6.7
2012-12-31                              84.5                             6.7
2013-12-31                              94.0                             6.9
2014-12-31                             113.3                             8.0
2015-12-31                              69.8                             6.4
2016-12-31                              39.8                             3.9
2017-12-31                              21.7                             2.5
2018-12-31                               7.0                             1.1
2019-12-31                               3.6                             0.5
2020-12-31                               8.8                             4.1

            Number of Observations  Definitive/Provisional Indicator
Date
1981-12-31                  3049.0                               1.0
1982-12-31                  3436.0                               1.0
1983-12-31                  4216.0                               1.0
1984-12-31                  5103.0                               1.0
1985-12-31                  5543.0                               1.0
1986-12-31                  5934.0                               1.0
1987-12-31                  6396.0                               1.0
1988-12-31                  6556.0                               1.0
1989-12-31                  6932.0                               1.0
1990-12-31                  7108.0                               1.0
1991-12-31                  6932.0                               1.0
1992-12-31                  7845.0                               1.0
1993-12-31                  8010.0                               1.0
1994-12-31                  8524.0                               1.0
1995-12-31                  8429.0                               1.0
1996-12-31                  7614.0                               1.0
1997-12-31                  7294.0                               1.0
1998-12-31                  6353.0                               1.0
1999-12-31                  6413.0                               1.0
2000-12-31                  5953.0                               1.0
2001-12-31                  6558.0                               1.0
2002-12-31                  6588.0                               1.0
2003-12-31                  7087.0                               1.0
2004-12-31                  6882.0                               1.0
2005-12-31                  7084.0                               1.0
2006-12-31                  6370.0                               1.0
2007-12-31                  6841.0                               1.0
2008-12-31                  6644.0                               1.0
2009-12-31                  6465.0                               1.0
2010-12-31                  6328.0                               1.0
2011-12-31                  6077.0                               1.0
```

```
2012-12-31                  5753.0                               1.0
2013-12-31                  5347.0                               1.0
2014-12-31                  5273.0                               1.0
2015-12-31                  8903.0                               1.0
2016-12-31                  9940.0                               1.0
2017-12-31                 11444.0                               1.0
2018-12-31                 12611.0                               1.0
2019-12-31                 12884.0                               1.0
2020-12-31                 14440.0                               1.0

Query with Sunspots > Mean:
            Yearly Mean Total Sunspot Number  Yearly Mean Standard Deviation  \
Date
1705-12-31                              96.7                             NaN
1717-12-31                             105.0                             NaN
1718-12-31                             100.0                             NaN
1726-12-31                             130.0                             NaN
1727-12-31                             203.3                             NaN
...                                      ...                             ...
2003-12-31                              99.3                             7.1
2011-12-31                              80.8                             6.7
2012-12-31                              84.5                             6.7
2013-12-31                              94.0                             6.9
2014-12-31                             113.3                             8.0

            Number of Observations  Definitive/Provisional Indicator
Date
1705-12-31                     NaN                               1.0
1717-12-31                     NaN                               1.0
1718-12-31                     NaN                               1.0
1726-12-31                     NaN                               1.0
1727-12-31                     NaN                               1.0
...                            ...                               ...
2003-12-31                  7087.0                               1.0
2011-12-31                  6077.0                               1.0
2012-12-31                  5753.0                               1.0
2013-12-31                  5347.0                               1.0
2014-12-31                  5273.0                               1.0

[136 rows x 4 columns]
```

In [14]:

```python
#Using the Quandl API, import the data and run the following descriptive stats where S
#Print the results of the describe function
#Print the count of observations
#Print the mad
#Print the mean
#Print the median
#Print the Max
#Print the Min
#Print the Mode
#Print the standard deviation
#Print the variance
#Print the Skewness


#load quandl, pd
import quandl
import pandas as pd
```

```
#api key kgECEhP4wEwUFD7MW_Ae
quandl.ApiConfig.api_key = "kgECEhP4wEwUFD7MW_Ae"

#sunsot data
dataset_code = "SIDC/SUNSPOTS_A"
data = quandl.get(dataset_code)

#sunspots is not equal to NaN
filtered_data = data[~data['Yearly Mean Total Sunspot Number'].isna()]

#results of the describe function
describe_results = filtered_data['Yearly Mean Total Sunspot Number'].describe()
print("describ function:")
print(describe_results)

#count of observations
count_observations = filtered_data['Yearly Mean Total Sunspot Number'].count()
print(f"count of observations: {count_observations}")

#MAD
mad = filtered_data['Yearly Mean Total Sunspot Number'].mad()
print(f"mean absolute deviation (MAD): {mad}")

#mean
mean = filtered_data['Yearly Mean Total Sunspot Number'].mean()
print(f"mean: {mean}")

#median
median = filtered_data['Yearly Mean Total Sunspot Number'].median()
print(f"median: {median}")

#max
max_value = filtered_data['Yearly Mean Total Sunspot Number'].max()
print(f"max: {max_value}")

#min
min_value = filtered_data['Yearly Mean Total Sunspot Number'].min()
print(f"min: {min_value}")

#mode
mode_value = filtered_data['Yearly Mean Total Sunspot Number'].mode()[0]
print(f"mode: {mode_value}")

#standard deviation
std_deviation = filtered_data['Yearly Mean Total Sunspot Number'].std()
print(f"standard deviation: {std_deviation}")

#variance
variance = filtered_data['Yearly Mean Total Sunspot Number'].var()
print(f"variance: {variance}")

#skewness
skewness = filtered_data['Yearly Mean Total Sunspot Number'].skew()
print(f"skewness: {skewness}")
```

```
descriptive statistics:
count    321.000000
mean      78.517134
std       62.091523
min        0.000000
25%       24.200000
50%       65.300000
75%      115.200000
max      269.300000
Name: Yearly Mean Total Sunspot Number, dtype: float64
count of observations: 321
mean absolute deviation (MAD): 51.02099552605273
mean: 78.51713395638629
median: 65.3
max: 269.3
min: 0.0
mode: 18.3
standard deviation: 62.09152256355228
variance: 3855.3571742601225
skewness: 0.8147812356121689
```

```
C:\Users\lexiw\AppData\Local\Temp\ipykernel_22936\3760674105.py:40: FutureWarning: Th
e 'mad' method is deprecated and will be removed in a future version. To compute the
same result, you may do `(df - df.mean()).abs().mean()`.
  mad = filtered_data['Yearly Mean Total Sunspot Number'].mad()
```

In [15]:

```
#Using the numpy random data generator, create a dataframe with the following columns
#Group the data by the weather column and then create a function to iterate through th
#Your function/variable that you created (weather_group) can be used for aggregation m
#Create another group – on Food (so you would have Weather and Food)
#Using the new groups – use the numpy function agg() to find the mean and median numbe



#load pd and np
import pandas as pd
import numpy as np

#make dataframe with weather, food price, and number
#set seed
np.random.seed(42)
data = {'Weather': np.random.choice(['Hot', 'Cold'], size=100),
    'Food Price': np.random.uniform(1, 10, size=100),
    'Number': np.random.randint(1, 100, size=100)}
df = pd.DataFrame(data)

#group by "weather"
weather_group = df.groupby('Weather')

#function to iterate through groups (first row,last row,mean)
def print_group_summary(group_name, group):
    print(f"group: {group_name}")
    print(f"first row:\n{group.head(1)}")
    print(f"last row:\n{group.tail(1)}")
    print(f"mean:\n{group.mean()}\n")
for group_name, group_data in weather_group:
    print_group_summary(group_name, group_data)

#food price group
food_group = df.groupby(['Weather', 'Food Price'])

#numpy's agg() function for mean, and median of number and food price
```

```
result = food_group.agg({'Number': ['mean', 'median'], 'Food Price': ['mean', 'median'
print("summary Statistics by weather and food price:")
print(result)
```

```
group: Cold
first row:
  Weather  Food Price  Number
1    Cold    7.976195       1
last row:
   Weather  Food Price  Number
98    Cold    1.463309      33
mean:
Food Price     5.435908
Number        50.339286
dtype: float64

group: Hot
first row:
  Weather  Food Price  Number
0     Hot    9.726262      97
last row:
   Weather  Food Price  Number
99     Hot    3.507818      14
mean:
Food Price     5.302751
Number        49.840909
dtype: float64

summary Statistics by weather and food price:
                   Number        Food Price          
                     mean median       mean    median
Weather Food Price                                   
Cold    1.049699     16.0   16.0   1.049699  1.049699
        1.228772     57.0   57.0   1.228772  1.228772
        1.331983     41.0   41.0   1.331983  1.331983
        1.463309     33.0   33.0   1.463309  1.463309
        1.572025     19.0   19.0   1.572025  1.572025
...                   ...    ...        ...       ...
Hot     9.367279     48.0   48.0   9.367279  9.367279
        9.455490     19.0   19.0   9.455490  9.455490
        9.486187     89.0   89.0   9.486187  9.486187
        9.726262     97.0   97.0   9.726262  9.726262
        9.881982     70.0   70.0   9.881982  9.881982

[100 rows x 4 columns]
```

```
C:\Users\lexiw\AppData\Local\Temp\ipykernel_22936\588814616.py:28: FutureWarning: The
default value of numeric_only in DataFrame.mean is deprecated. In a future version, i
t will default to False. In addition, specifying 'numeric_only=None' is deprecated. S
elect only valid columns or specify the value of numeric_only to silence this warnin
g.
  print(f"mean:\n{group.mean()}\n")
C:\Users\lexiw\AppData\Local\Temp\ipykernel_22936\588814616.py:28: FutureWarning: The
default value of numeric_only in DataFrame.mean is deprecated. In a future version, i
t will default to False. In addition, specifying 'numeric_only=None' is deprecated. S
elect only valid columns or specify the value of numeric_only to silence this warnin
g.
  print(f"mean:\n{group.mean()}\n")
```

In [17]:

```
#Using the dataframe you created in #5, select the first 3 rows
#Using the concat function from pandas, put the 3 rows you selected back with the orig
```

```
#Using the append function, take those 3 rows and the last 2 rows of the original Data

#load pd, np
import pandas as pd
import numpy as np

#random data
#make dataframe with weather, food price, and number
#set seed
np.random.seed(42)
data = {'Weather': np.random.choice(['Hot', 'Cold'], size=100),
    'Food Price': np.random.uniform(1, 10, size=100),
    'Number': np.random.randint(1, 100, size=100)}
df = pd.DataFrame(data)

#first 3 rows
first_3_rows = df.head(3)

# Concatenate 3 rows
concatenated_df = pd.concat([df, first_3_rows])

#concatenated DataFrame
print("concatenated dataframe:")
print(concatenated_df)

#last 2 rows
last_2_rows = df.tail(2)

#append func
appended_df = first_3_rows.append(last_2_rows)
print("\nAppended dataframe:")
print(appended_df)
```

```
concatenated dataframe:
   Weather  Food Price  Number
0      Hot    9.726262      97
1     Cold    7.976195       1
2      Hot    9.455490      19
3      Hot    9.053446       2
4      Hot    6.381100      53
..     ...         ...     ...
98    Cold    1.463309      33
99     Hot    3.507818      14
0      Hot    9.726262      97
1     Cold    7.976195       1
2      Hot    9.455490      19

[103 rows x 3 columns]

Appended dataframe:
   Weather  Food Price  Number
0      Hot    9.726262      97
1     Cold    7.976195       1
2      Hot    9.455490      19
98    Cold    1.463309      33
99     Hot    3.507818      14
```

```
C:\Users\lexiw\AppData\Local\Temp\ipykernel_22936\829514449.py:32: FutureWarning: The
frame.append method is deprecated and will be removed from pandas in a future versio
n. Use pandas.concat instead.
  appended_df = first_3_rows.append(last_2_rows)
```

In [18]:

```
#Using the two CSV files dest.csv and tips.csv, we will bring together two datasets, c
#Using the merge() function, bring dest and tips together on the “EmpNr” column and pr
#Using the join() function, query both files and print the results

#load pd
import pandas as pd

#load dest csv.csv
#load tips csv.csv
dest_df = pd.read_csv("dest csv.csv")
tips_df = pd.read_csv("tips csv.csv")


#merge
merged_df = pd.merge(dest_df, tips_df, on="EmpNr")
print("merged dataframe:")
print(merged_df)
```

```
merged dataframe:
   EmpNr       Dest  Amount
0      5  The Hague    10.0
1      9  Rotterdam     5.0
```

In [19]:

```
#Using the two CSV files dest.csv and tips.csv, we will bring together two datasets, c
#Using the merge() function, bring dest and tips together on the “EmpNr” column and pr
#Using the join() function, query both files and print the results

#load pd
import pandas as pd

#load dest csv.csv
#load tips csv.csv
dest_df = pd.read_csv("dest csv.csv")
tips_df = pd.read_csv("tips csv.csv")

#"EmpNr" column as index
dest_df.set_index("EmpNr", inplace=True)
tips_df.set_index("EmpNr", inplace=True)

#join them
joined_df = dest_df.join(tips_df, how="inner")

#reset index
joined_df.reset_index(inplace=True)
print("Joined DataFrame:")
print(joined_df)
```

```
Joined DataFrame:
   EmpNr       Dest  Amount
0      5  The Hague    10.0
1      9  Rotterdam     5.0
```

In [20]:

```
#Using the WHO_first9cols.csv file, select the first 3 rows, including the headers for
#Check for missing values
#Count the number of NaN values
#Print any non-missing values
#Replace the missing values with a scalar value

#WHO_first9cols.CSV
```

```
#load pd
import pandas as pd

#WHO_first9cols.CSV
df = pd.read_csv("WHO_first9cols.CSV")

#3 rows, headers, country, net, male %
selected_df = df[['Country', 'Net primary school enrolment ratio male (%)']].head(3)

#missing values
missing_values = selected_df.isna()

#NaN values
num_nan_values = missing_values.sum().sum()

#non-missing values
non_missing_values = selected_df.dropna()
print("Non-Missing Values:")
print(non_missing_values)

#missing value w/ scalar value
selected_df.fillna(0, inplace=True)
print("\nDataFrame with Missing Values Replaced:")
print(selected_df)
print(f"\nNumber of NaN Values: {num_nan_values}")
```

```
Non-Missing Values:
   Country  Net primary school enrolment ratio male (%)
1  Albania                                         94.0
2  Algeria                                         96.0

DataFrame with Missing Values Replaced:
       Country  Net primary school enrolment ratio male (%)
0  Afghanistan                                          0.0
1      Albania                                         94.0
2      Algeria                                         96.0

Number of NaN Values: 1
```

In [ ]: