**COIMBATORE INSTITUTE OF TECHNOLOGY**

**1. EXPLORATORY DATA ANALYSIS ON THE GIVEN DATA SETS (COUNTRY-WISE-AVERAGE.CSV AND MALNUTRITION-ESTIMATES.CSV)**

- Nandhakumar T(1832035)

**AIM:**

To perform exploratory data analysis without using built-in packages and to visualize the results.

**DESCRIPTION:**

The given data(**country-wise-average.csv and malnutrition-estimates.csv**) involves many missing values. Also some of the data are not in usable format. For that the types of the data to be changed(column). After changing types, missing values should be filled with appropriate values such that, for categorical values we should choose mode, for numerical data mean should be used(not skewed). After filling missing values we have to determine the summary statistics for the given data to get an overview about the data. For that each and every function (mean, median, mode, quartiles(25%), quartile(75%)) should be written in scratch. After performing all the above steps finally the results should be visualized.

**CODE:**

**#importing basic packages to load file**

import numpy as np

import pandas as pd

import seaborn as sns

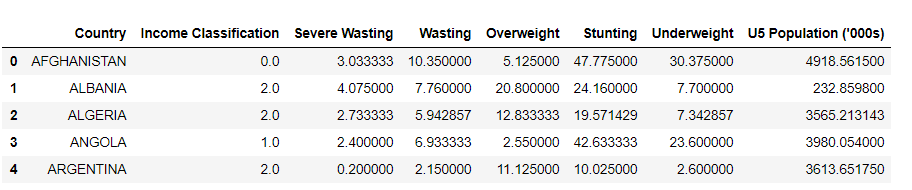
**#load data**

df\_country = pd.read\_csv(r'C:/Users/THANGAVEL/Desktop/country-wise-average.csv')

df\_mal = pd.read\_csv(r'C:/Users/THANGAVEL/Desktop/malnutrition-estimates.csv')

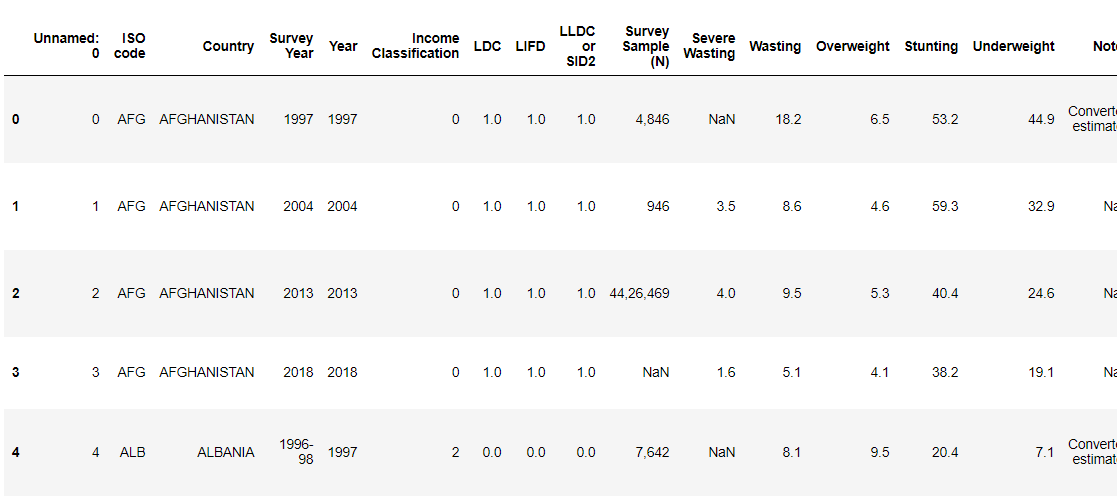
df\_country.head()

**OUTPUT:**



df\_mal.head()

**OUTPUT:**

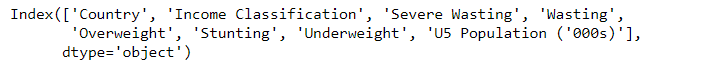


**#EXPLORATORY DATA ANALYSIS**

**#columns**

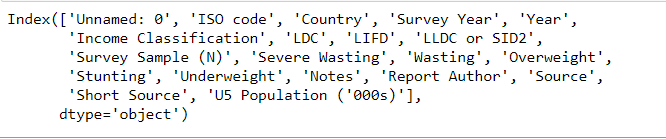
df\_country.columns

**OUTPUT :**



df\_mal.columns

**OUTPUT:**

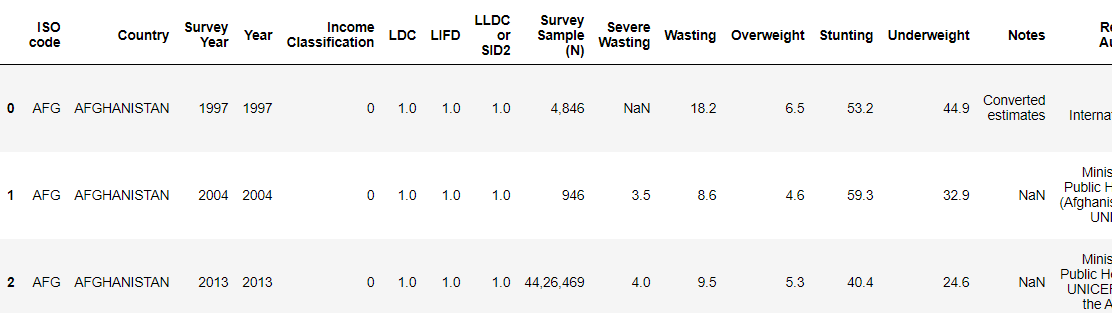


**#removing first colums**

df\_mal = df\_mal.iloc[:,1:]

df\_mal.head()

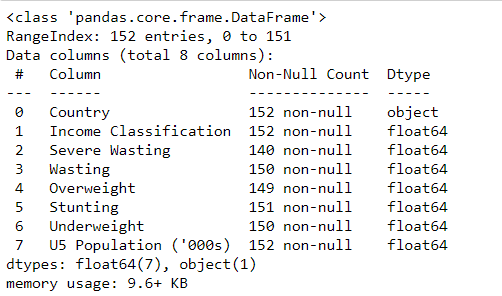
**OUTPUT:**



**#country\_average info**

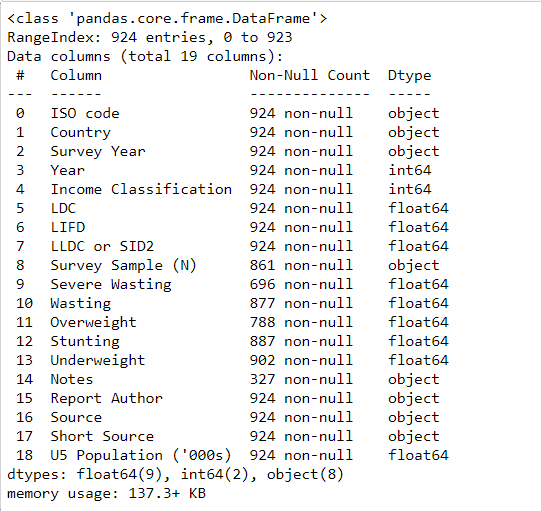
df\_country.info()

**OUTPUT:**



df\_mal.info()

**OUTPUT:**



**#mean,median,mode functions for EDA**

def find\_mean(df,x):

s = 0

for i in df:

s+=i

print("MEAN of %s : %.3f"%(x,(s/len(df))))

def find\_stdev(df,x):

s = 0

for i in df:

s+=i

m = s/len(df)

total = 0

for i in df:

total = total+((i-m)\*\*2)

res = (total/len(df))\*\*(1/2)

print("STDEV of %s : %.3f"%(x,res))

def find\_median(df,x):

df1 = df.sort\_values(ascending = True)

mid = (len(df)+1)//2

print("MEDIAN of %s : %d"%(x,df1[mid]))

def find\_mode(df,x):

dic = {}

for i in df:

if i not in dic:

dic[i] = 1

else:

dic[i] += 1

res = max(dic.values())

print("MODE of %s : %d"%(x,res))

def find\_quart(df,x,pos):

df1 = df.sort\_values(ascending = True)

qar = pos\*((len(df)+1)//4)

print("QUARTILE of %s : %d"%(x,df1[qar]))

**#filling missing values in country averages(df\_country)**

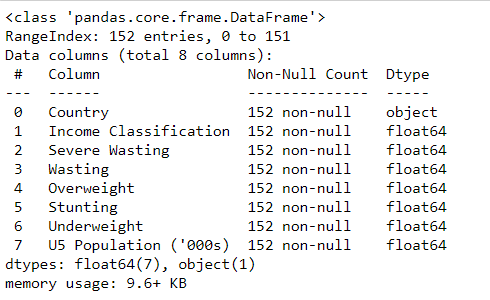
for i in df\_country.columns:

if type(df\_country[i][0]) == np.float64:

df\_country[i]=df\_country[i].fillna(df\_country[i].mean())

df\_country.info()

**OUTPUT:**



**#filling missing values in malnutrition (df\_mal)**

df\_mal = df\_mal.fillna(df\_mal.mean())

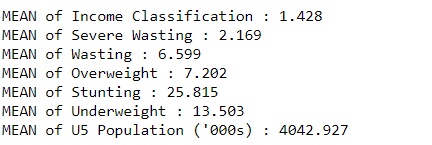
**#finding mean for all possible rows(int and float) on country\_average**

for i in df\_country.columns:

if type(df\_country[i][0]) == np.float64:

find\_mean(df\_country[i],i)

**OUTPUT:**



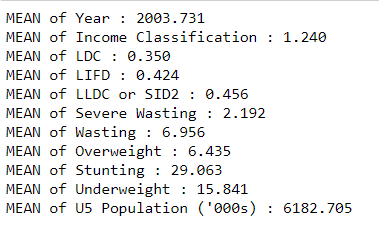
**#finding mean for all possible rows(int and float) in malnutrition**

for i in df\_mal.columns:

if (type(df\_mal[i][1]) == np.float64) or (type(df\_mal[i][1]) == np.int64):

find\_mean(df\_mal[i],i)

**OUTPUT:**



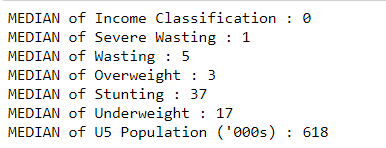
**#finding median for all possible rows(int and float) on country\_average**

for i in df\_country.columns:

if type(df\_country[i][0]) == np.float64:

find\_median(df\_country[i],i)

**OUTPUT:**



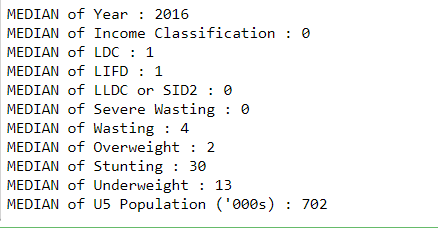
**#finding median for all possible rows(int and float) in malnutrition**

for i in df\_mal.columns:

if (type(df\_mal[i][1]) == np.float64) or (type(df\_mal[i][1]) == np.int64):

find\_median(df\_mal[i],i)

**OUPUT:**



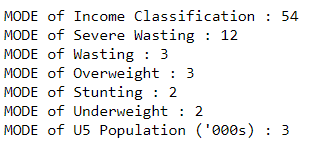
**#finding mean for all possible rows(int and float) on country\_average**

for i in df\_country.columns:

if type(df\_country[i][0]) == np.float64:

find\_mode(df\_country[i],i)

**OUTPUT:**



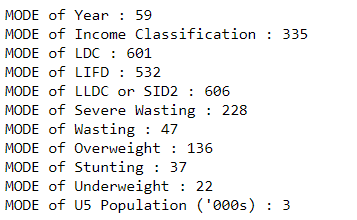
**#finding mode for all possible rows(int and float) in malnutrition**

for i in df\_mal.columns:

if (type(df\_mal[i][1]) == np.float64) or (type(df\_mal[i][1]) == np.int64):

find\_mode(df\_mal[i],i)

**OUTPUT:**



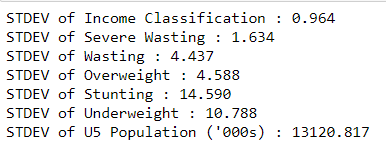
**#finding standard deviation for all possible rows(int and float) on country\_average**

for i in df\_country.columns:

if type(df\_country[i][0]) == np.float64:

find\_stdev(df\_country[i],i)

**OUTPUT:**



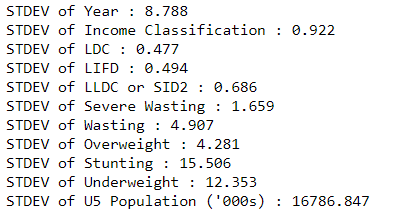
**#finding standard deviation for all possible rows(int and float) in malnutrition**

for i in df\_mal.columns:

if (type(df\_mal[i][1]) == np.float64) or (type(df\_mal[i][1]) == np.int64):

find\_stdev(df\_mal[i],i)

**OUTPUT:**



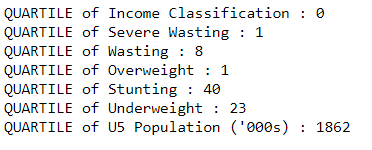
**#finding 1st quartile(75%) for all possible rows(int and float) on country\_average**

for i in df\_country.columns:

if type(df\_country[i][0]) == np.float64:

find\_quart(df\_country[i],i,1)

**OUTPUT:**



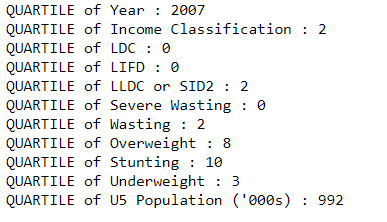
**#finding 1st quartile(75%) for all possible rows(int and float) in malnutrition**

for i in df\_mal.columns:

if (type(df\_mal[i][1]) == np.float64) or (type(df\_mal[i][1]) == np.int64):

find\_quart(df\_mal[i],i,1)

**OUTPUT:**



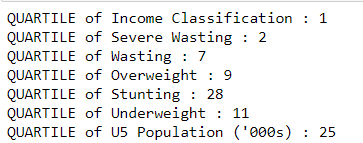
**#finding 1st quartile(25%) for all possible rows(int and float) on country\_average**

for i in df\_country.columns:

if type(df\_country[i][0]) == np.float64:

find\_quart(df\_country[i],i,3)

**OUTPUT:**



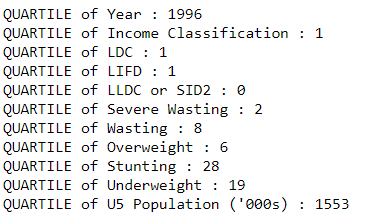
**#finding 1st quartile(25%) for all possible rows(int and float) in malnutrition**

for i in df\_mal.columns:

if (type(df\_mal[i][1]) == np.float64) or (type(df\_mal[i][1]) == np.int64):

find\_quart(df\_mal[i],i,3)

**OUTPUT:**

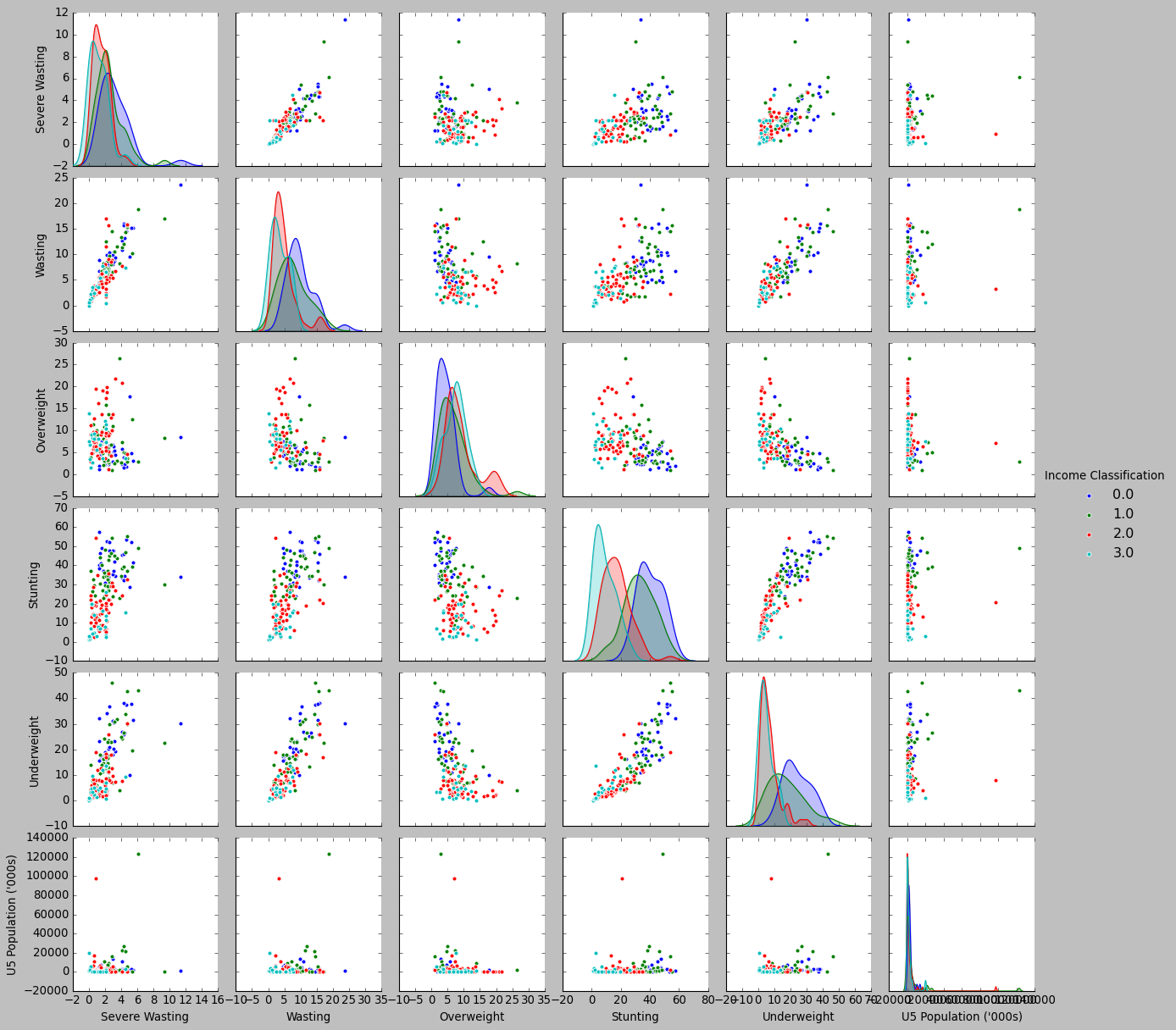


**#VISUALIZATION**

**#pairplot for country Income Classification**

sns.pairplot(df\_country, hue='Income Classification');

**OUTPUT:**



**#bar plot**

ax = sns.barplot(x="Country", y="Income Classification", data=df\_country)

**OUTPUT:**

