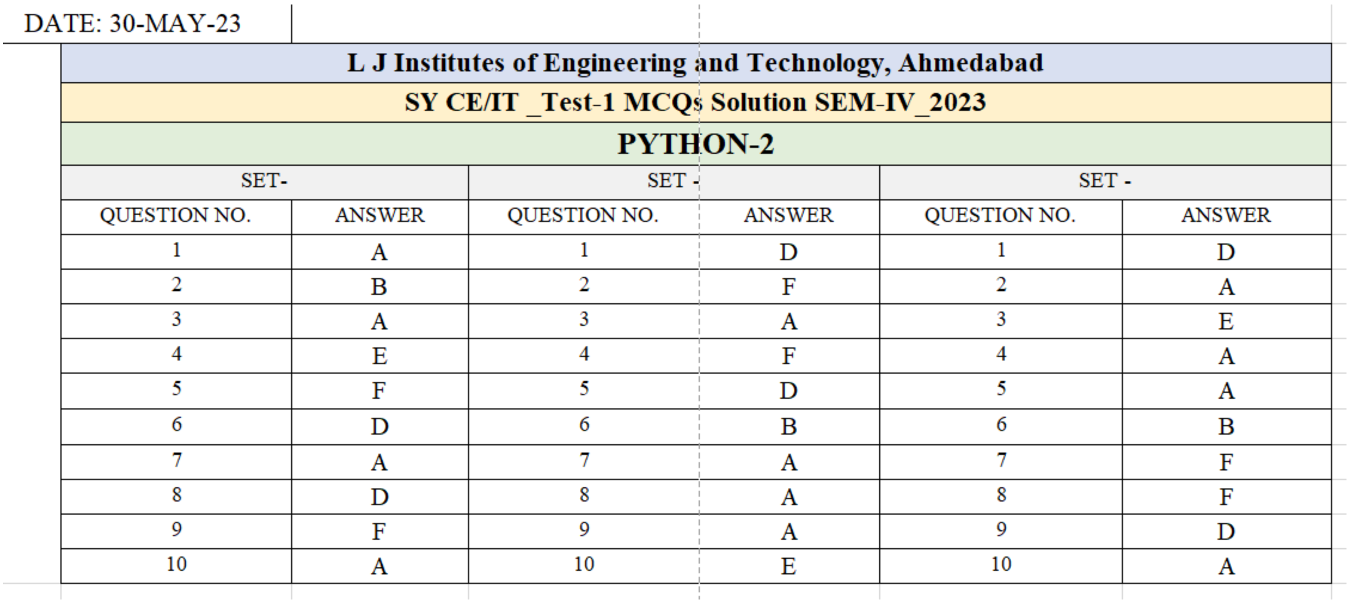


DATE: 30-MAY-23

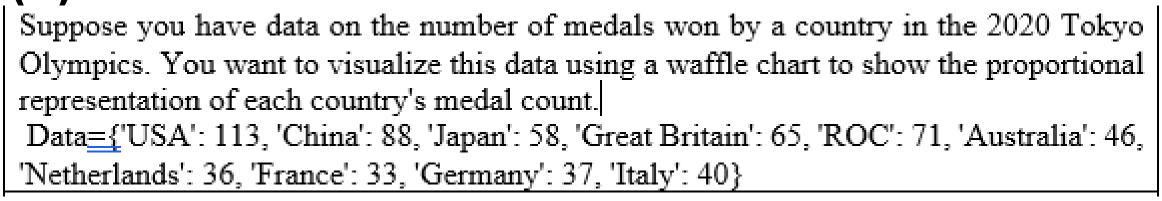
L J Institutes of Engineering and Technology, Ahmedabad

SY CE/IT _Test-1 MCQs Solution SEM-IV_2023

PYTHON-2

SET-		SET -		SET -	
QUESTION NO.	ANSWER	QUESTION NO.	ANSWER	QUESTION NO.	ANSWER
1	A	1	D	1	D
2	B	2	F	2	A
3	A	3	A	3	E
4	E	4	F	4	A
5	F	5	D	5	A
6	D	6	B	6	B
7	A	7	A	7	F
8	D	8	A	8	F
9	F	9	A	9	D
10	A	10	E	10	A

Q-2 (1)

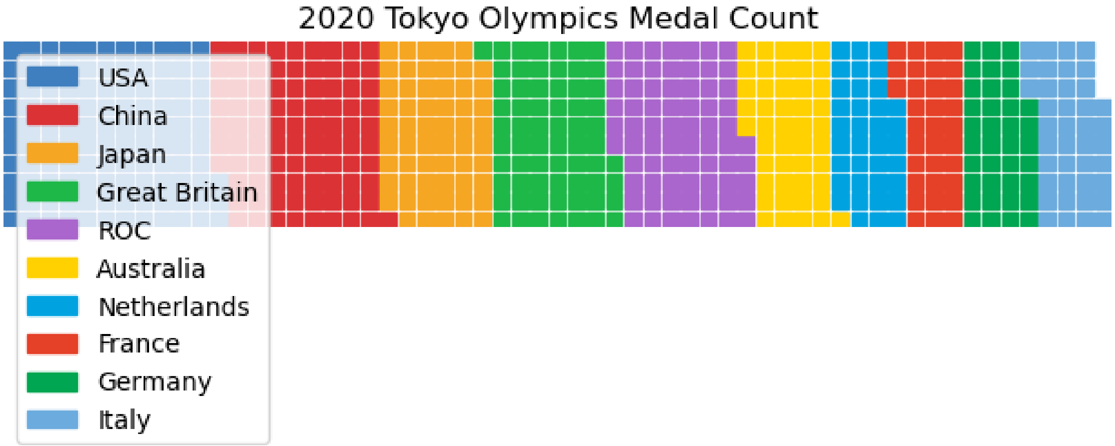


Suppose you have data on the number of medals won by a country in the 2020 Tokyo Olympics. You want to visualize this data using a waffle chart to show the proportional representation of each country's medal count.

Data={'USA': 113, 'China': 88, 'Japan': 58, 'Great Britain': 65, 'ROC': 71, 'Australia': 46, 'Netherlands': 36, 'France': 33, 'Germany': 37, 'Italy': 40}

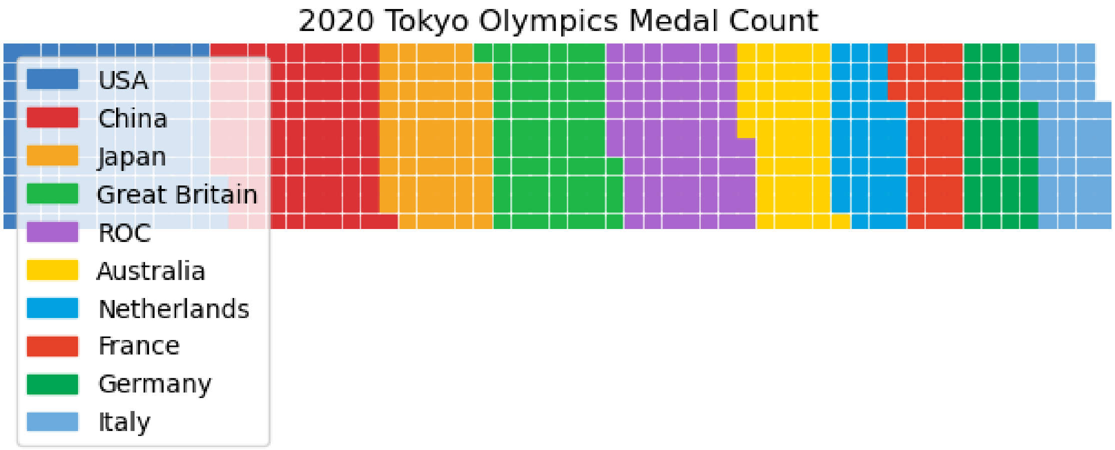
In [1]:

```python
# Import necessary libraries
import pandas as pd
import matplotlib.pyplot as plt
from pywaffle import Waffle
# Create a DataFrame from the given data
data = pd.DataFrame.from_dict({'USA': 113, 'China': 88, 'Japan': 58,
'Great Britain': 65, 'ROC': 71,
'Australia': 46, 'Netherlands': 36,
'France': 33, 'Germany': 37, 'Italy': 40},
orient='index', columns=['medal_count'])
# Set up waffle chart parameters
fig = plt.figure(
FigureClass=Waffle,
rows=10,
values=data['medal_count'],
labels=list(data.index),
colors=['#3F7FBF', '#DB3236', '#F5A623', '#1EB849', '#AA66CC',
'#FFD100', '#00A3E0', '#E54028', '#00A651', '#6CABDD'],
legend={'loc': 'upper left'}
)
# Add title
plt.title('2020 Tokyo Olympics Medal Count')
# Show the chart
plt.show()
```



In [2]:

```python
import pandas as pd
import matplotlib.pyplot as plt
from pywaffle import Waffle

# Create a list of countries and their corresponding medal counts
countries = ['USA', 'China', 'Japan', 'Great Britain', 'ROC', 'Australia', 'Netherla
medal_counts = [113, 88, 58, 65, 71, 46, 36, 33, 37, 40]

# Create a DataFrame from the countries and medal counts
data = pd.DataFrame({'countries': countries, 'medal_counts': medal_counts})

# Set up waffle chart parameters
fig = plt.figure(
FigureClass=Waffle,#------------0.5 MARK FOR THIS LINE
rows=10,
values=data['medal_counts'],#--------0.5 MARK FOR THIS LINE
labels=list(data.countries),#-------------0.5 MARK FOR THIS LINE
colors=['#3F7FBF', '#DB3236', '#F5A623', '#1EB849', '#AA66CC',
'#FFD100', '#00A3E0', '#E54028', '#00A651', '#6CABDD'],
legend={'loc': 'upper left'}
)
# Add title
plt.title('2020 Tokyo Olympics Medal Count')
# Show the chart
plt.show()
```



Q-2 (2)

- You have been hired as a network analyst by a company to analyze the social network of their employees. The company has provided you with the following data: There are 5 employees in the company, each identified by a unique ID from 1 to 5.

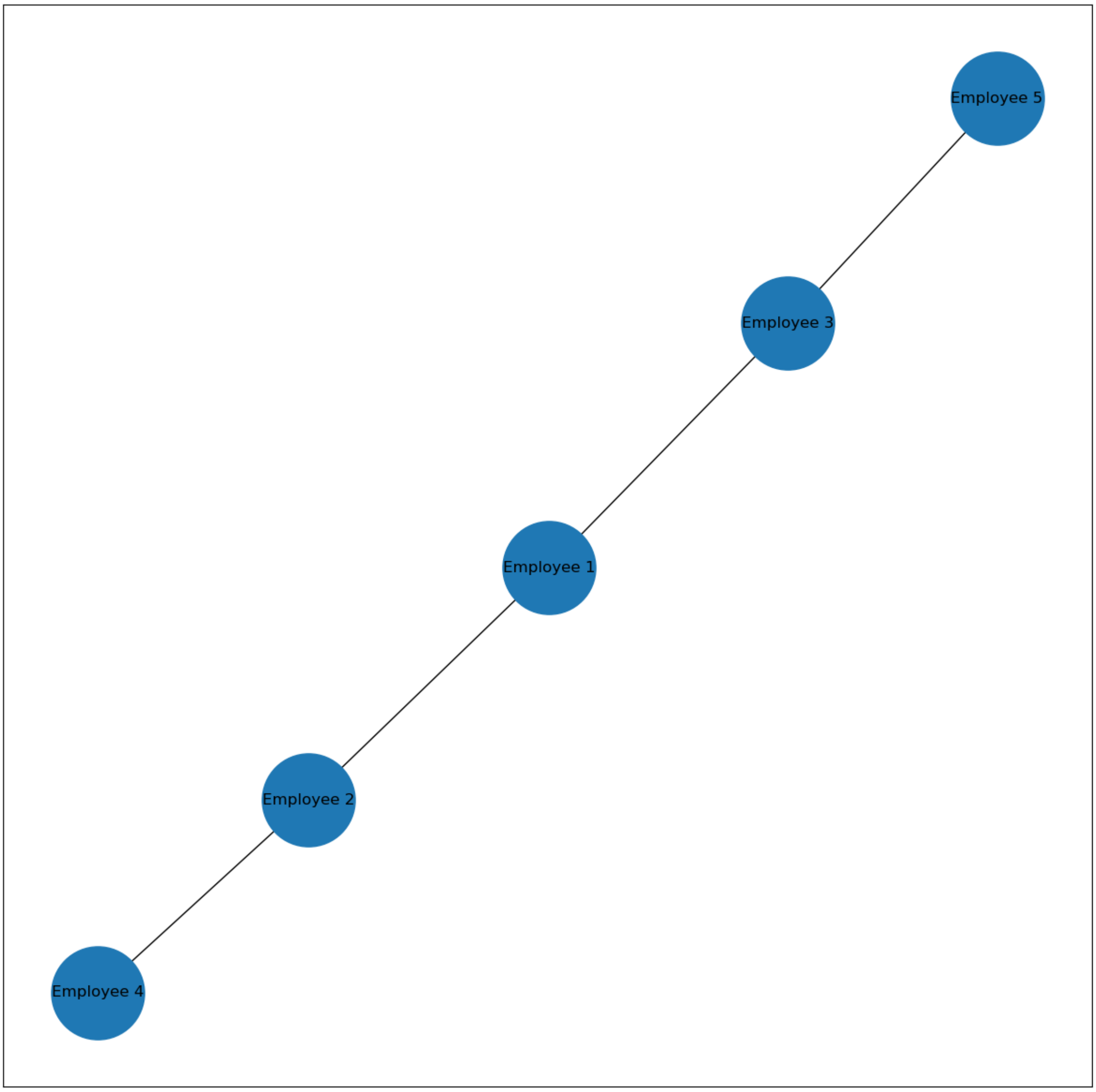
  The following relationships exist between the employees:

  1. Employee 1 is friends with Employee 2 and Employee 3.
  2. Employee 2 is friends with Employee 4.
  3. Employee 3 is friends with Employee 5.

Your task is to create a NetworkX graph representing this social network and display it.

In [3]:

```python
import networkx as nx
import matplotlib.pyplot as plt
# Create an empty undirected graph
G = nx.Graph()#nx.DiGraph()
plt.figure(figsize=(15,15))
# Add nodes to the graph
G.add_nodes_from([1, 2, 3, 4, 5])
# Add edges to the graph
G.add_edge(1, 2)
G.add_edge(1, 3)
G.add_edge(2, 4)
G.add_edge(3, 5)
# Set the node labels
labels = {1: 'Employee 1', 2: 'Employee 2',
3: 'Employee 3',
4: 'Employee 4', 5: 'Employee 5'}
# Draw the graph with node labels
nx.draw_networkx(G, labels=labels, with_labels=True, node_size=5100)
plt.show()
```



Q-3 MARK DISTRIBUTION

1-10: EACH RIGHT Answer GIVES A 0.5 MARKS

11-14: EACH RIGHT Answer GIVES A 0.5 MARKS

In [4]:

```
# Load in some packages
import numpy as np
import pandas as pd
import os
```

1. Make a data frame with the variable name df

In [5]:

```
df=pd.read_csv("diabetes_unclean.csv")
```

2. To display the specific statistics or measures that are relevant for object-type columns

In [6]:

```
# display the specific statistics or measures that are relevant for object-type colu
df.describe(include=object)
```

Out[6]:

	Gender	CLASS
count	1009	1009
unique	3	5
top	M	Y
freq	570	840

3. To display the specific statistics or measures that are relevant for numerical-type columns

In [7]:

```
df.describe()
```

Out[7]:

	ID	No_Pation	AGE	Urea	Cr	HbA1c	
count	1009.000000	1.009000e+03	1008.000000	1008.000000	1007.000000	1006.000000	1007
mean	339.161546	2.717448e+05	53.620040	5.131094	68.973188	8.284155	4
std	239.738169	3.365681e+06	8.740975	2.931136	59.813297	2.533576	1
min	1.000000	1.230000e+02	25.000000	0.500000	6.000000	0.900000	0
25%	127.000000	2.406500e+04	51.000000	3.700000	48.000000	6.500000	4
50%	296.000000	3.439900e+04	55.000000	4.600000	60.000000	8.000000	4
75%	548.000000	4.539000e+04	59.000000	5.700000	73.000000	10.200000	5
max	800.000000	7.543566e+07	79.000000	38.900000	800.000000	16.000000	10

4. How many rows and columns are in a given dataset

In [8]:

```
print("number of rows",df.shape[0])
```

```
number of rows 1009
```

In [9]:

```
print("number of columns",df.shape[1])
```

```
number of columns 14
```

5. To check the missing values

In [10]:

```
#To check the missing values
df.isnull().sum()
```

Out[10]:

```
ID           0
No_Pation    0
Gender       0
AGE          1
Urea         1
Cr           2
HbA1c        3
Chol         2
TG           2
HDL          1
LDL          2
VLDL         1
BMI          0
CLASS        0
dtype: int64
```

6. To replace the missing values in the column "HbA1c" with their mean value

In [11]:

```
#replace the missing values in the column "HbA1c" with it's mean
df['HbA1c']=df['HbA1c'].fillna(df['HbA1c'].mean())
```

In [12]:

```
#To confirm to change
df.isnull().sum()
```

Out[12]:

```
ID           0
No_Pation    0
Gender       0
AGE          1
Urea         1
Cr           2
HbA1c        0
Chol         2
TG           2
HDL          1
LDL          2
VLDL         1
BMI          0
CLASS        0
dtype: int64
```

7. Dropping the missing values of other columns

In [13]:

```
# Dropping the missing values of other columns
df=df.dropna()
df.isna().sum()
```

Out[13]:

```
ID           0
No_Pation    0
Gender       0
AGE          0
Urea         0
Cr           0
HbA1c        0
Chol         0
TG           0
HDL          0
LDL          0
VLDL         0
BMI          0
CLASS        0
dtype: int64
```

8. To check the information on the dataset

In [14]:

```
#check the information of dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 997 entries, 0 to 1008
Data columns (total 14 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   ID         997 non-null    int64  
 1   No_Pation  997 non-null    int64  
 2   Gender     997 non-null    object 
 3   AGE        997 non-null    float64
 4   Urea       997 non-null    float64
 5   Cr         997 non-null    float64
 6   HbA1c      997 non-null    float64
 7   Chol       997 non-null    float64
 8   TG         997 non-null    float64
 9   HDL        997 non-null    float64
 10  LDL        997 non-null    float64
 11  VLDL       997 non-null    float64
 12  BMI        997 non-null    float64
 13  CLASS      997 non-null    object 
dtypes: float64(10), int64(2), object(2)
memory usage: 116.8+ KB
```

9. in a class column "N " and "Y " replace with "N" and "Y" respectively. And check specific statistics or measures that are relevant for object-type columns

In [15]:

```
df.CLASS.unique()
```

Out[15]:

```
array(['N', 'N ', 'P', 'Y', 'Y '], dtype=object)
```

In [16]:

```
#in a class columns "N " and "Y " replace with "N" and "Y" respectively
df['CLASS'].loc[df['CLASS']=="N "]="N"
df['CLASS'].loc[df['CLASS']=="Y "]="Y"
```

```
C:\Users\VISHAL\AppData\Local\Temp\ipykernel_17836\2208156168.py:3: Settin
gWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  df['CLASS'].loc[df['CLASS']=="Y "]="Y"
```

In [17]:

```
df.CLASS.unique()
```

Out[17]:

```
array(['N', 'P', 'Y'], dtype=object)
```

10. display the correlation between variables

In [18]:

```
#show the correlation between variables?
df.corr()
```

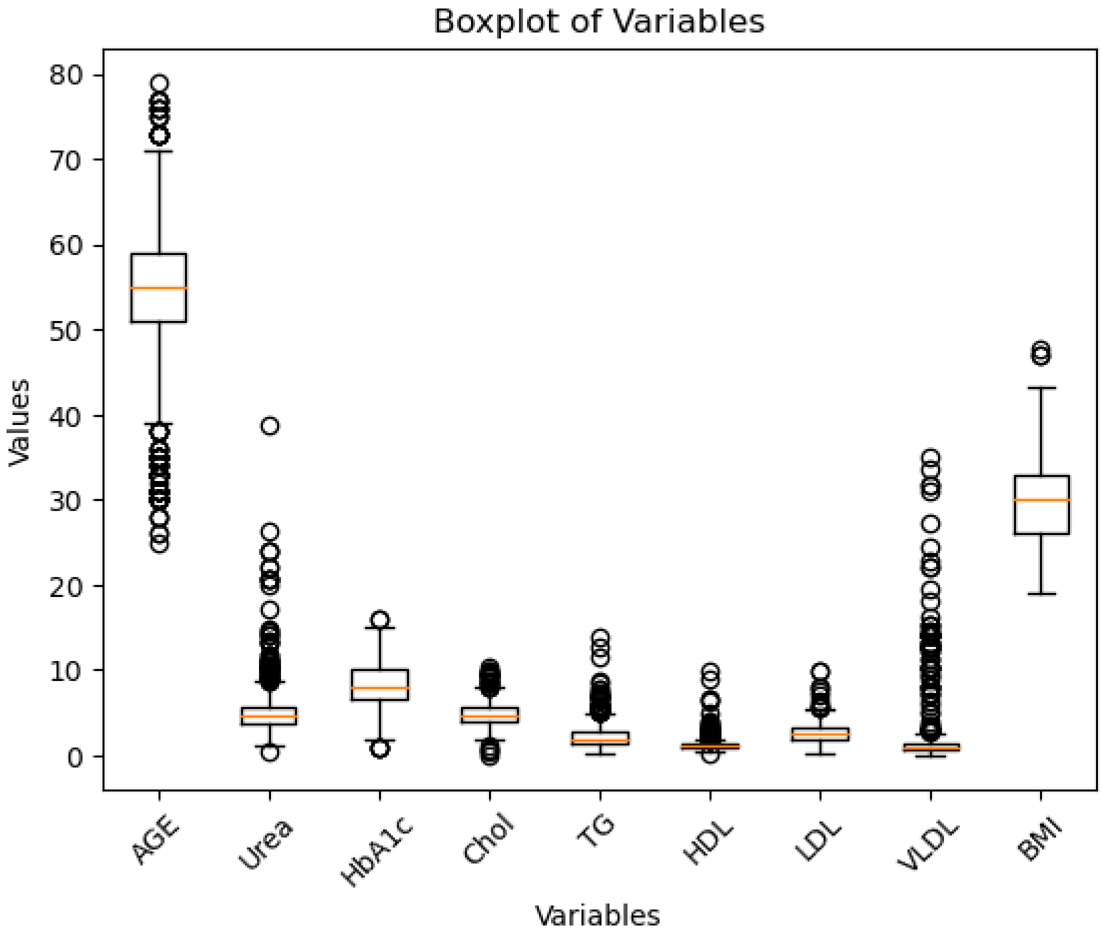
Out[18]:

	ID	No_Pation	AGE	Urea	Cr	HbA1c	Chol	
ID	1.000000	0.064599	-0.072687	-0.094891	-0.100046	-0.009037	0.045414	-0.054
No_Pation	0.064599	1.000000	-0.088870	-0.019061	0.000973	-0.032350	-0.030288	-0.039
AGE	-0.072687	-0.088870	1.000000	0.108613	0.056940	0.384675	0.038966	0.149
Urea	-0.094891	-0.019061	0.108613	1.000000	0.624810	-0.023307	0.001286	0.040
Cr	-0.100046	0.000973	0.056940	0.624810	1.000000	-0.037735	-0.007636	0.056
HbA1c	-0.009037	-0.032350	0.384675	-0.023307	-0.037735	1.000000	0.177676	0.214
Chol	0.045414	-0.030288	0.038966	0.001286	-0.007636	0.177676	1.000000	0.318
TG	-0.054110	-0.039859	0.149274	0.040939	0.056031	0.214030	0.318894	1.000
HDL	0.025226	-0.013554	-0.021029	-0.037843	-0.023578	0.030455	0.103370	-0.083
LDL	-0.065718	-0.003520	0.011496	-0.006673	0.040981	0.011536	0.419237	0.015
VLDL	0.145700	0.113635	-0.090796	-0.011573	0.010328	0.072641	0.076373	0.145
BMI	0.047270	0.017738	0.381176	0.045753	0.054847	0.413130	0.016989	0.110

11 checking the outliers in the dataset for the following parameters: 'AGE', 'Urea', 'HbA1c', 'Chol', 'TG', 'HDL', 'LDL', 'VLDL', 'BMI' using box plot with labels and title

In [19]:

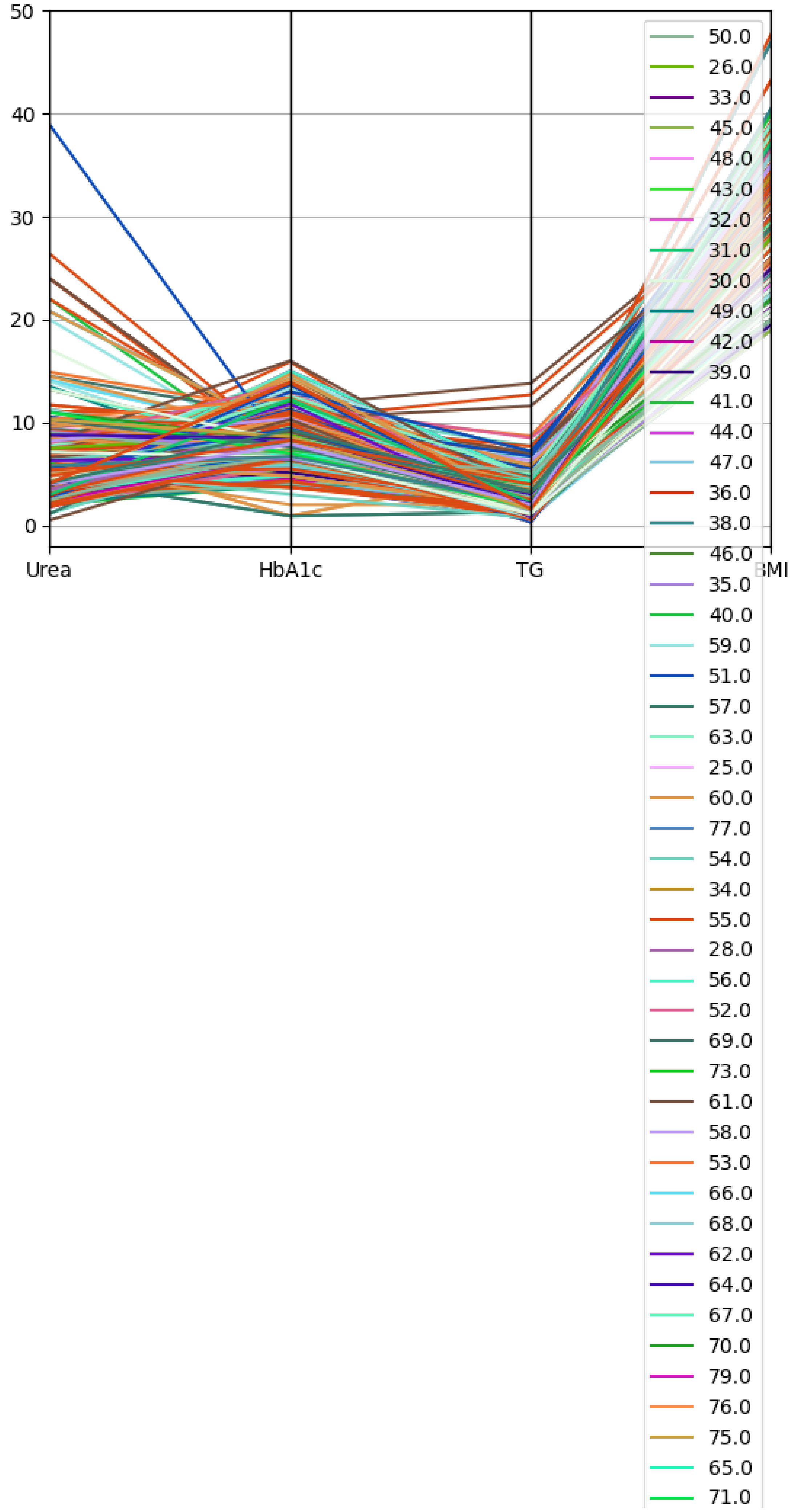
```
#checking the outliers in dataset
import matplotlib.pyplot as plt

columns = ['AGE', 'Urea', 'HbA1c', 'Chol', 'TG', 'HDL', 'LDL', 'VLDL', 'BMI']
plt.boxplot(df[columns])
plt.xlabel('Variables')
plt.ylabel('Values')
plt.title('Boxplot of Variables')
plt.xticks(range(1, len(columns) + 1), columns, rotation=45)
plt.show()

```



12. Visualized the "Urea", "HbA1c", "TG" and "BMI" parameters for different ages using parallel_coordinates with labels and title

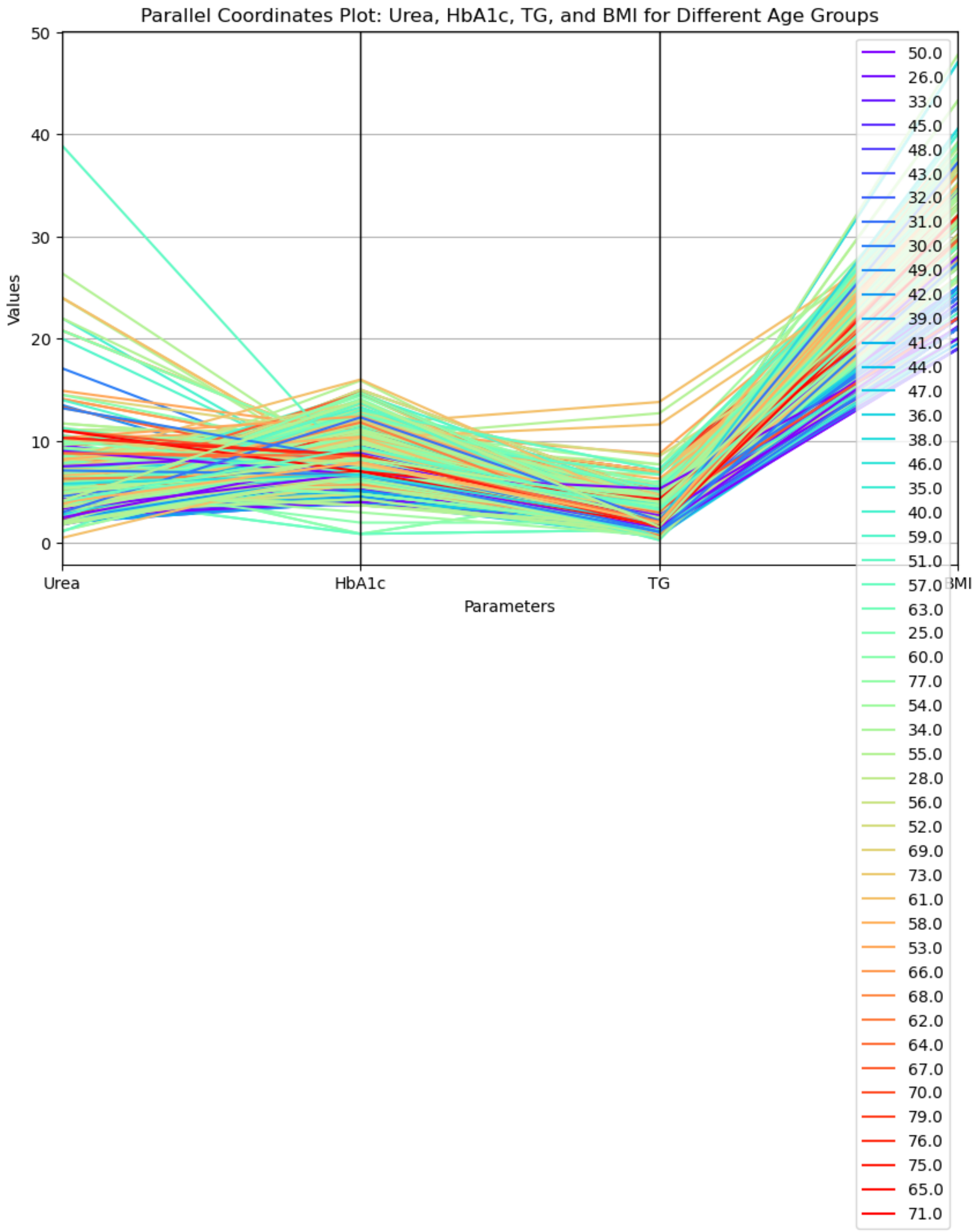
In [20]:

```
# visulize the "Urea","HbA1c","TG" and "BMI" parmaters for different age using paral
import matplotlib.pyplot as plt
fig,ax=plt.subplots()
pd.plotting.parallel_coordinates(df,'AGE',["Urea","HbA1c","TG","BMI"])
plt.show()
```


50
40
30
20
10
0
Urea
HbA1c
TG
BMI
50.0
26.0
33.0
45.0
48.0
43.0
32.0
31.0
30.0
49.0
42.0
39.0
41.0
44.0
47.0
36.0
38.0
46.0
35.0
40.0
59.0
51.0
57.0
63.0
25.0
60.0
77.0
54.0
34.0
55.0
28.0
56.0
52.0
69.0
73.0
61.0
58.0
53.0
66.0
68.0
62.0
64.0
67.0
70.0
79.0
76.0
75.0
65.0
71.0

In [21]:

```
import pandas as pd
import matplotlib.pyplot as plt
from pandas.plotting import parallel_coordinates

# Selecting the specific columns of interest
df_selected = df[['AGE', 'Urea', 'HbA1c', 'TG', 'BMI']]

# Creating the parallel coordinates plot
plt.figure(figsize=(10, 6))  # Adjust the figure size as needed
parallel_coordinates(df_selected, 'AGE', colormap='rainbow')

# Adding labels and a title to the plot
plt.xlabel('Parameters')
plt.ylabel('Values')
plt.title('Parallel Coordinates Plot: Urea, HbA1c, TG, and BMI for Different Age Gro

# Displaying the plot
plt.show()

```



13. Remove the rows whose gender column has an “f” value and give the frequency count of the “F” and “M” values in different CLASS values

In [22]:

```
df[df["Gender"]=="f"]
```

Out[22]:

	ID	No_Pation	Gender	AGE	Urea	Cr	HbA1c	Chol	TG	HDL	LDL	VLDL	BMI
991	195	4543	f	55.0	4.1	34.0	13.9	5.4	1.6	1.6	3.1	0.7	33.0
1008	195	4543	f	55.0	4.1	34.0	13.9	5.4	1.6	1.6	3.1	0.7	33.0

In [23]:

```
print(df.shape)
df=df[df["Gender"]!="f"]
print(df.shape)
```

```
(997, 14)
(995, 14)
```

OR

In [24]:

```
print(df.shape)
df=df.drop(df[df["Gender"]=="f"].index)
print(df.shape)
```

```
(995, 14)
(995, 14)
```

In [25]:

```
print("F in class value N")
print(len(df[(df["Gender"]=="F") & (df["CLASS"]=="N")]))
print("F in class value P")
print(len(df[(df["Gender"]=="F") & (df["CLASS"]=="P")]))
print("F in class value Y")
print(len(df[(df["Gender"]=="F") & (df["CLASS"]=="Y")]))
print("M in class value N")
print(len(df[(df["Gender"]=="M") & (df["CLASS"]=="N")]))
print("M in class value P")
print(len(df[(df["Gender"]=="M") & (df["CLASS"]=="P")]))
print("M in class value Y")
print(len(df[(df["Gender"]=="M") & (df["CLASS"]=="Y")]))
```

```
F in class value N
64
F in class value P
17
F in class value Y
351
M in class value N
39
M in class value P
36
M in class value Y
488
```

In [26]:

```
print("F in class value N")
print(df[(df["Gender"]=="F") & (df["CLASS"]=="N")].shape[0])
print("F in class value P")
print(df[(df["Gender"]=="F") & (df["CLASS"]=="P")].shape[0])
print("F in class value Y")
print(df[(df["Gender"]=="F") & (df["CLASS"]=="Y")].shape[0])
print("M in class value N")
print(df[(df["Gender"]=="M") & (df["CLASS"]=="N")].shape[0])
print("M in class value P")
print(df[(df["Gender"]=="M") & (df["CLASS"]=="P")].shape[0])
print("M in class value Y")
print(df[(df["Gender"]=="M") & (df["CLASS"]=="Y")].shape[0])
```

```
F in class value N
64
F in class value P
17
F in class value Y
351
M in class value N
39
M in class value P
36
M in class value Y
488
```

In [27]:

```
F_N=0
F_P=0
F_Y=0
M_N=0
M_P=0
M_Y=0
for i,j in zip((df.Gender),(df.CLASS)):
    if i=="F" and j=="N":
        F_N+=1
    if i=="F" and j=="P":
        F_P+=1
    if i=="F" and j=="Y":
        F_Y+=1
    if i=="M" and j=="N":
        M_N+=1
    if i=="M" and j=="P":
        M_P+=1
    if i=="M" and j=="Y":
        M_Y+=1

print("F in class value N")
print(F_N)
print("F in class value P")
print(F_P)
print("F in class value Y")
print(F_Y)
print("M in class value N")
print(M_N)
print("M in class value P")
print(M_P)
print("M in class value Y")
print(M_Y)
```

```
F in class value N
64
F in class value P
17
F in class value Y
351
M in class value N
39
M in class value P
36
M in class value Y
488
```

In [28]:

```
pd.crosstab(df["Gender"],df["CLASS"])
```

Out[28]:

CLASS	**N**	**P**	**Y**
Gender			
F	64	17	351
M	39	36	488

14. remove the outliers in the "HbA1c" columns and print the shape of the data frame

In [29]:

```
# remove the outliers in "HbA1c" columns
import pandas as pd

# Assuming 'df' is your DataFrame and 'column_name' is the name of the column with o

# Calculate the IQR (Interquartile Range) for the column
Q1 = df["HbA1c"].quantile(0.25)
Q3 = df["HbA1c"].quantile(0.75)
IQR = Q3 - Q1

# Set the threshold for identifying outliers
threshold = 1.5

# Filter the DataFrame to exclude rows with outliers
df_without_outliers = df[(df["HbA1c"] >= Q1 - threshold * IQR) & (df["HbA1c"] <= Q3

```

In [30]:

```
df_without_outliers.shape
```

Out[30]:

```
(989, 14)
```