T1_Practice Book_VHA

26. Create a Pandas DataFrame from the following table and write code to remove all rows from this table containing at least one NaN value

```
import pandas as pd
import numpy as np

# Data from the table, with NaN values for missing entries
data = [["Emma", "North", 50000.0, np.nan, np.nan, 0.0],
        ["Sofia", "East", 420.0, 380.0, np.nan, np.nan],
        ["Marku", "West", 72.0, 3.0, np.nan, np.nan],
        ["Edward", "West", 49.0, 42.0, np.nan, np.nan],
        ["Thomas", "South", np.nan, np.nan, np.nan, np.nan],
        ["Ethan", np.nan, np.nan, np.nan, np.nan, np.nan],
        ["Arun", "West", 67000.0, 39000.0, np.nan, np.nan],
        ["Anika", "East", 65000.0, 45000.0, np.nan, np.nan],
        ["Paulo", "South", 67000.0, np.nan, np.nan, np.nan]]

# Create DataFrame, including column names
df = pd.DataFrame(data, columns=["name", "region", "50000.0", "0.0", "expe", "0.0"])

# Drop rows with all NaN values
df = df.dropna(how='any')

print(df)
```

```
Empty DataFrame
Columns: [name, region, 50000.0, 0.0, expe, 0.0]
Index: []
```

27. Create a Pandas DataFrame from the following table and write code to remove all rows from this table only if all of their values are NaN

```
import pandas as pd
import numpy as np

# Data from the table, with NaN values for missing entries
data = [["Emma", "North", 50000.0, np.nan, np.nan, 0.0],
        ["Sofia", "East", 420.0, 380.0, np.nan, np.nan],
        ["Marku", "West", 72.0, 3.0, np.nan, np.nan],
        ["Edward", "West", 49.0, 42.0, np.nan, np.nan],
        ["Thomas", "South", np.nan, np.nan, np.nan, np.nan],
        ["Ethan", np.nan, np.nan, np.nan, np.nan, np.nan],
        ["Arun", "West", 67000.0, 39000.0, np.nan, np.nan],
        ["Anika", "East", 65000.0, 45000.0, np.nan, np.nan],
        ["Paulo", "South", 67000.0, np.nan, np.nan, np.nan]]

# Create DataFrame, including column names
df = pd.DataFrame(data, columns=["name", "region", "50000.0", "0.0", "expe", "0.0"])

# Drop rows with all NaN values
df = df.dropna(how='all')

print(df)
```

```
      name region  50000.0      0.0  expe  0.0
0    Emma  North  50000.0      NaN   NaN  0.0
1   Sofia   East    420.0    380.0   NaN  NaN
2   Marku   West     72.0      3.0   NaN  NaN
3  Edward   West     49.0     42.0   NaN  NaN
4  Thomas  South      NaN      NaN   NaN  NaN
5   Ethan    NaN      NaN      NaN   NaN  NaN
6    Arun   West  67000.0  39000.0   NaN  NaN
7   Anika   East  65000.0  45000.0   NaN  NaN
8   Paulo  South  67000.0      NaN   NaN  NaN
```

28. Create a Pandas DataFrame from the following table and write code to drop all columns containing NaN

```
import pandas as pd
import numpy as np

# Data from the table, with NaN values for missing entries
data = [["Emma", "North", 50000.0, np.nan, np.nan, 0.0],
        ["Sofia", "East", 420.0, 380.0, np.nan, np.nan],
        ["Marku", "West", 72.0, 3.0, np.nan, np.nan],
        ["Edward", "West", 49.0, 42.0, np.nan, np.nan],
        ["Thomas", "South", np.nan, np.nan, np.nan, np.nan],
        ["Ethan", np.nan, np.nan, np.nan, np.nan, np.nan],
        ["Arun", "West", 67000.0, 39000.0, np.nan, np.nan],
        ["Anika", "East", 65000.0, 45000.0, np.nan, np.nan],
        ["Paulo", "South", 67000.0, np.nan, np.nan, np.nan]]

# Create DataFrame, including column names
df = pd.DataFrame(data, columns=["name", "region", "50000.0", "0.0", "expe", "0.0"])

# Drop rows with all NaN values
df = df.dropna(how='any',axis=1)

print(df)
print("shape of dataframe",df.shape)
```

```
      name
0    Emma
1   Sofia
2   Marku
3  Edward
4  Thomas
5   Ethan
6    Arun
7   Anika
8   Paulo
shape of dataframe (9, 1)
```

29 Write Python code to remove outliers from any given DataFrame.

```
data = {'Name': ['William', 'Emma', 'Sofia', 'Markus',
'Edward','Thomas','Ethan',np.nan,'Arun','Anika','Paulo'],
'Region': [np.nan,'North','East',np.nan,'West',
'West', 'South',np.nan,'West','East', 'South'],
'Sales': [50000.0, 52000.0, np.nan,np.nan,42000.0,
72000.0,49000.0,np.nan,67000.0,65000.0,67000.0],
'Expenses': [42000.0, 43000.0,np.nan,np.nan, 38000.0,
390000.0,42000.0,np.nan,39000.0,50000.0,45000.0]}
# Create the DataFrame
df = pd.DataFrame(data)
print(df.shape)
def remove_outliers(df, col):
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]
# Remove outliers from column Sales using the remove_outliers function
df_no_outliers = remove_outliers(df, 'Sales')
print(df_no_outliers.shape)
df_no_outliers
```

```
(11, 4)
(8, 4)
```

	Name	Region	Sales	Expenses
0	William	NaN	50000.0	42000.0
1	Emma	North	52000.0	43000.0
4	Edward	West	42000.0	38000.0
5	Thomas	West	72000.0	390000.0
6	Ethan	South	49000.0	42000.0
8	Arun	West	67000.0	39000.0
9	Anika	East	65000.0	50000.0
10	Paulo	South	67000.0	45000.0

30 Consider the following data:

data = {

"A": ["TeamA", "TeamB", "TeamB", "TeamC", "TeamA"],

"B": [50, 40, 40, 30, 50],

"C": [True, False, False, False, True]

}

Convert this to a Pandas DataFrame and remove duplicate rows from it. Reset index values.

```
import pandas as pd

# Create a DataFrame from the data
data = {
  "A": ["TeamA", "TeamB", "TeamB", "TeamC", "TeamA"],
  "B": [50, 40, 40, 30, 50],
  "C": [True, False, False, False, True]
}

df = pd.DataFrame(data)

# Remove duplicate rows and reset index
df = df.drop_duplicates().reset_index(drop=True)

# Print the DataFrame
print(df)

df.set_index('A',inplace=True)
df
```

```
        A   B      C
0  TeamA  50   True
1  TeamB  40  False
2  TeamC  30  False
```

A	B	C
TeamA	50	True
TeamB	40	False
TeamC	30	False

31 Consider the following autompg dataset:

https://raw.githubusercontent.com/Jovita7/Data-Analysis-and-Visualization/main/auto-mpg.csv Write Python code to convert it to a DataFrame and remove mpg and cylinders columns from it

```
import pandas as pd

# URL of the dataset
url = 'https://raw.githubusercontent.com/Jovita7/Data-Analysis-and-Visualization/main/auto-mpg.csv'

# Load the dataset into a DataFrame
df = pd.read_csv(url)

# Display the first few rows of the DataFrame to understand its structure
print("Original DataFrame:")
print(df.head())
```

```
Original DataFrame:
    mpg  cylinders  displacement horsepower  weight  acceleration  model year  \
0  18.0          8         307.0        130    3504          12.0          70
1  15.0          8         350.0        165    3693          11.5          70
2  18.0          8         318.0        150    3436          11.0          70
3  16.0          8         304.0        150    3433          12.0          70
4  17.0          8         302.0        140    3449          10.5          70

   origin                   car name
0       1  chevrolet chevelle malibu
1       1          buick skylark 320
2       1         plymouth satellite
3       1              amc rebel sst
4       1                ford torino
```

```
# Remove the 'mpg' and 'cylinders' columns
df = df.drop(columns=['mpg', 'cylinders'])

# Display the first few rows of the updated DataFrame
print("\nDataFrame after removing 'mpg' and 'cylinders' columns:")
print(df.head())
```

```
DataFrame after removing 'mpg' and 'cylinders' columns:
   displacement horsepower  weight  acceleration  model year  origin  \
0         307.0        130    3504          12.0          70       1
1         350.0        165    3693          11.5          70       1
2         318.0        150    3436          11.0          70       1
3         304.0        150    3433          12.0          70       1
4         302.0        140    3449          10.5          70       1
```

```
                       car name
0  chevrolet chevelle malibu
1          buick skylark 320
2         plymouth satellite
3              amc rebel sst
4                ford torino
```

32. Use the file heights_weights.csv (https://raw.githubusercontent.com/Jovita7/Data-Analysis-and-Visualization/main/heights_weights.csv) which contains 10000 non-null values for heights and weights. The Male column shows 1 if the person is a Male and 0 if the person is a Female.

1. Convert this file into a pandas Data Frame. (0.5 marks)
2. Display basic information like memory and data types for this data frame. (0.5 marks)
3. Display basic statistics like mean, std, quartiles, etc. for this data frame. (0.5 marks)
4. Create a correlation table for the data frame and comment about what kind of correlation is there between Height and Weight. (0.5 marks)
5. Do Height and Weight contain any outliers? (1 mark)

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
import pandas as pd

# URL of the dataset
url = 'https://raw.githubusercontent.com/Jovita7/Data-Analysis-and-Visualization/main/heights_weights.csv'

# Load the dataset into a DataFrame
df = pd.read_csv(url)

# Display the first few rows of the DataFrame to verify the loading
print("Original DataFrame:")
print(df.head())
```

```
Original DataFrame:
      Height      Weight  Male
0  73.847017  241.893563     1
1  68.781904  162.310473     1
2  74.110105  212.740856     1
3  71.730978  220.042470     1
4  69.881796  206.349801     1
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
# Display basic information about the DataFrame
print("\nBasic Information:")
print(df.info())
```

```
Basic Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Height  10000 non-null  float64
 1   Weight  10000 non-null  float64
 2   Male    10000 non-null  int64
dtypes: float64(2), int64(1)
memory usage: 234.5 KB
None
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
# Display basic statistics for the DataFrame
print("\nBasic Statistics:")
print(df.describe())
```

```
Basic Statistics:
             Height        Weight          Male
count  10000.000000  10000.000000  10000.000000
mean      66.367560    161.440357      0.500000
std        3.847528     32.108439      0.500025
min       54.263133     64.700127      0.000000
25%       63.505620    135.818051      0.000000
50%       66.318070    161.212928      0.500000
75%       69.174262    187.169525      1.000000
max       78.998742    269.989699      1.000000
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
# Create a correlation table
correlation_matrix = df.corr()

# Display the correlation table
print("\nCorrelation Table:")
print(correlation_matrix)

# Comment about the correlation between Height and Weight
height_weight_corr = correlation_matrix.loc['Height', 'Weight']
print(f"\nCorrelation between Height and Weight: {height_weight_corr}")
if height_weight_corr > 0:
    print("There is a positive correlation between Height and Weight.")
elif height_weight_corr < 0:
    print("There is a negative correlation between Height and Weight.")
else:
    print("There is no correlation between Height and Weight.")
```

```
Correlation Table:
          Height    Weight      Male
Height  1.000000  0.924756  0.691072
Weight  0.924756  1.000000  0.796723
Male    0.691072  0.796723  1.000000

Correlation between Height and Weight: 0.9247562987409196
There is a positive correlation between Height and Weight.
```

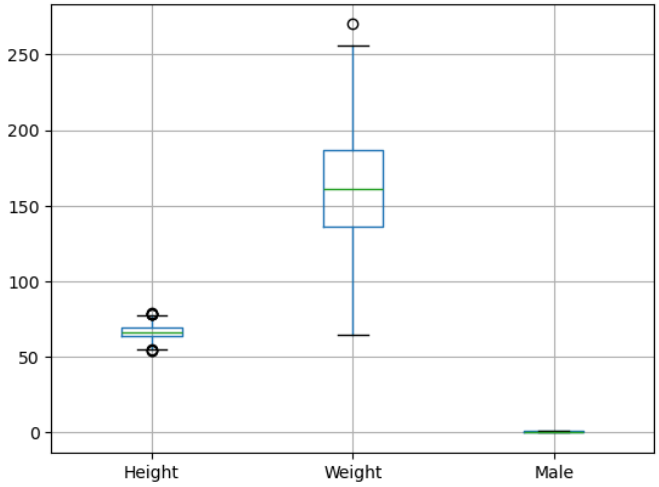
Unsupported Cell Type. Double-Click to inspect/edit the content.


```
# Function to detect outliers using the IQR method
def detect_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = df[(df[column] < lower_bound) | (df[column] > upper_bound)]
    return outliers

# Detect outliers in Height
height_outliers = detect_outliers(df, 'Height')
print(f"\nNumber of outliers in Height: {len(height_outliers)}")

# Detect outliers in Weight
weight_outliers = detect_outliers(df, 'Weight')
print(f"Number of outliers in Weight: {len(weight_outliers)}")

# Display some of the outliers if they exist
if not height_outliers.empty:
    print("\nOutliers in Height:")
    print(height_outliers.head())

if not weight_outliers.empty:
    print("\nOutliers in Weight:")
    print(weight_outliers.head())
```

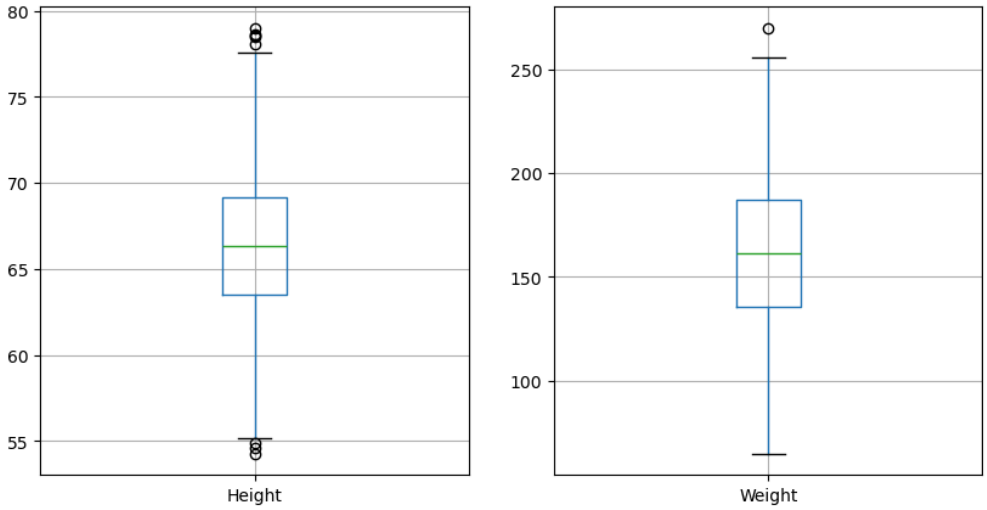
```
Number of outliers in Height: 8
Number of outliers in Weight: 1

Outliers in Height:
          Height      Weight  Male
994   78.095867  255.690835     1
1317  78.462053  227.342565     1
2014  78.998742  269.989699     1
3285  78.528210  253.889004     1
3757  78.621374  245.733783     1

Outliers in Weight:
          Height      Weight  Male
2014  78.998742  269.989699     1
```

```
df.boxplot()
```

```
<Axes: >
```




```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(1, 2, figsize=(10,5))
df.boxplot(column=['Height'], ax=ax[0])
df.boxplot(column=['Weight'], ax=ax[1])
plt.show()
```



33 Use the file ipl-matches.csv which contains data of all the IPL matches from year 2008 to 2022. Read this csv file and display the basic information like memory and data types for this data frame. Write python code for the following cases:

1. List out all matches gone in superover.
2. How Many Matches won by Chennai Super Kings at Kolkata.
3. In How Many Matches MS Dhoni is Player of Match Vs Mumbai Indians.
4. Display list of all matches in which Gujarat Titans won the Toss and Elected to Bat and won the match.
5. Display list of all matches won by Gujarat Titans.

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
import pandas as pd

# Load the dataset into a DataFrame
df = pd.read_csv("ipl-matches.csv")

# Display basic information about the DataFrame
print("\nBasic Information:")
print(df.info())
```

```
Basic Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 950 entries, 0 to 949
Data columns (total 20 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
```

```
 0   ID              950 non-null    int64
 1   City            899 non-null    object
 2   Date            950 non-null    object
 3   Season          950 non-null    object
 4   MatchNumber     950 non-null    object
 5   Team1           950 non-null    object
 6   Team2           950 non-null    object
 7   Venue           950 non-null    object
 8   TossWinner      950 non-null    object
 9   TossDecision    950 non-null    object
 10  SuperOver       946 non-null    object
 11  WinningTeam     946 non-null    object
 12  WonBy           950 non-null    object
 13  Margin          932 non-null    float64
 14  method          19 non-null     object
 15  Player_of_Match 946 non-null    object
 16  Team1Players    950 non-null    object
 17  Team2Players    950 non-null    object
 18  Umpire1         950 non-null    object
 19  Umpire2         950 non-null    object
dtypes: float64(1), int64(1), object(18)
memory usage: 148.6+ KB
None
```

```
# Display the first few rows of the DataFrame to understand its structure
print("\nFirst few rows of the DataFrame:")
print(df.head())
```

```
First few rows of the DataFrame:
        ID       City        Date Season  MatchNumber  \
0  1312200  Ahmedabad  2022-05-29   2022        Final
1  1312199  Ahmedabad  2022-05-27   2022  Qualifier 2
2  1312198    Kolkata  2022-05-25   2022   Eliminator
3  1312197    Kolkata  2022-05-24   2022  Qualifier 1
4  1304116     Mumbai  2022-05-22   2022           70

                         Team1                Team2  \
0            Rajasthan Royals        Gujarat Titans
1  Royal Challengers Bangalore      Rajasthan Royals
2  Royal Challengers Bangalore  Lucknow Super Giants
3            Rajasthan Royals        Gujarat Titans
4          Sunrisers Hyderabad          Punjab Kings

                             Venue            TossWinner TossDecision  \
0  Narendra Modi Stadium, Ahmedabad      Rajasthan Royals          bat
1  Narendra Modi Stadium, Ahmedabad      Rajasthan Royals        field
2            Eden Gardens, Kolkata  Lucknow Super Giants        field
3            Eden Gardens, Kolkata        Gujarat Titans        field
4          Wankhede Stadium, Mumbai   Sunrisers Hyderabad          bat

  SuperOver                 WinningTeam    WonBy  Margin method  \
0         N               Gujarat Titans  Wickets     7.0    NaN
1         N            Rajasthan Royals  Wickets     7.0    NaN
2         N  Royal Challengers Bangalore     Runs    14.0    NaN
3         N               Gujarat Titans  Wickets     7.0    NaN
4         N                 Punjab Kings  Wickets     5.0    NaN

  Player_of_Match                                       Team1Players  \
0       HH Pandya  ['YBK Jaiswal', 'JC Buttler', 'SV Samson', 'D ...
1      JC Buttler  ['V Kohli', 'F du Plessis', 'RM Patidar', 'GJ ...
2      RM Patidar  ['V Kohli', 'F du Plessis', 'RM Patidar', 'GJ ...
3       DA Miller  ['YBK Jaiswal', 'JC Buttler', 'SV Samson', 'D ...
4    Harpreet Brar  ['PK Garg', 'Abhishek Sharma', 'RA Tripathi', ...

                                     Team2Players        Umpire1  \
0  ['WP Saha', 'Shubman Gill', 'MS Wade', 'HH Pan...    CB Gaffaney
1  ['YBK Jaiswal', 'JC Buttler', 'SV Samson', 'D ...    CB Gaffaney
2  ['Q de Kock', 'KL Rahul', 'M Vohra', 'DJ Hooda...  J Madanagopal
3  ['WP Saha', 'Shubman Gill', 'MS Wade', 'HH Pan...   BNJ Oxenford
4  ['JM Bairstow', 'S Dhawan', 'M Shahrukh Khan',...   AK Chaudhary

         Umpire2
0    Nitin Menon
1    Nitin Menon
2       MA Gough
3      VK Sharma
4  NA Patwardhan
```

```
# Filter matches where SuperOver is 1
super_over_matches = df[df['SuperOver'] == "Y"]
print("super_over_matches",super_over_matches.shape[0])

# Display the matches gone in super over
print("\nMatches gone in super over:")
print(super_over_matches[['ID', 'Team1', 'Team2', 'City', 'Date']])
```

```
super_over_matches 14

Matches gone in super over:
          ID                     Team1                     Team2  \
114  1254077             Delhi Capitals         Sunrisers Hyderabad
158  1216512      Kolkata Knight Riders         Sunrisers Hyderabad
159  1216517             Mumbai Indians             Kings XI Punjab
184  1216547  Royal Challengers Bangalore              Mumbai Indians
```

```
192  1216493              Delhi Capitals             Kings XI Punjab
203  1178426              Mumbai Indians          Sunrisers Hyderabad
244  1175365       Kolkata Knight Riders              Delhi Capitals
339  1082625               Gujarat Lions              Mumbai Indians
474   829741            Rajasthan Royals             Kings XI Punjab
533   729315       Kolkata Knight Riders            Rajasthan Royals
608   598017  Royal Challengers Bangalore            Delhi Daredevils
621   598004         Sunrisers Hyderabad  Royal Challengers Bangalore
819   419121         Chennai Super Kings             Kings XI Punjab
883   392190       Kolkata Knight Riders            Rajasthan Royals

          City        Date
114    Chennai  2021-04-25
158  Abu Dhabi  2020-10-18
159        NaN  2020-10-18
184        NaN  2020-09-28
192        NaN  2020-09-20
203     Mumbai  2019-05-02
244      Delhi  2019-03-30
339     Rajkot  2017-04-29
474  Ahmedabad  2015-04-21
533  Abu Dhabi  2014-04-29
608  Bangalore  2013-04-16
621  Hyderabad  2013-04-07
819    Chennai  2010-03-21
883  Cape Town  2009-04-23
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
# Filter matches won by Chennai Super Kings at Kolkata
csk_kolkata_wins = df[(df['WinningTeam'] == 'Chennai Super Kings') & (df['City'] == 'Kolkata')]

# Count the number of matches
csk_kolkata_wins_count = csk_kolkata_wins.shape[0]
print(f"\nNumber of matches won by Chennai Super Kings at Kolkata: {csk_kolkata_wins_count}")
```

```
Number of matches won by Chennai Super Kings at Kolkata: 5
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
# Filter matches where MS Dhoni was the Player of the Match and Mumbai Indians were one of the teams
dhoni_pom_vs_mi = df[(df['Player_of_Match'] == 'MS Dhoni') &
                     ((df['Team1'] == 'Mumbai Indians') | (df['Team2'] == 'Mumbai Indians'))]

# Count the number of such matches
dhoni_pom_vs_mi_count = dhoni_pom_vs_mi.shape[0]
print(f"\nNumber of matches where MS Dhoni is Player of Match Vs Mumbai Indians: {dhoni_pom_vs_mi_count}")
dhoni_pom_vs_mi
```

```
Number of matches where MS Dhoni is Player of Match Vs Mumbai Indians: 1
```

	ID	City	Date	Season	MatchNumber	Team1	Team2	Venue	TossWinner	TossDecision	SuperOver	WinningTeam	WonBy	Margin
630	548379	Bangalore	2012-05-23	2012	Elimination Final	Chennai Super Kings	Mumbai Indians	M Chinnaswamy Stadium	Mumbai Indians	field	N	Chennai Super Kings	Runs	38.0

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
# Filter matches where Gujarat Titans won the toss, elected to bat, and won the match
gt_toss_bat_wins = df[(df['TossWinner'] == 'Gujarat Titans') &
                      (df['TossDecision'] == 'bat') &
                      (df['WinningTeam'] == 'Gujarat Titans')]

# Display the filtered matches
print("\nMatches where Gujarat Titans won the Toss, Elected to Bat, and won the match:")
print(gt_toss_bat_wins[['ID', 'Team1', 'Team2', 'City', 'Date']])
```

```
Matches where Gujarat Titans won the Toss, Elected to Bat, and won the match:
         ID           Team1                 Team2         City        Date
17  1304103  Gujarat Titans   Lucknow Super Giants         Pune  2022-05-10
39  1304081  Gujarat Titans  Kolkata Knight Riders  Navi Mumbai  2022-04-23
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
# Filter matches where Gujarat Titans won the toss, elected to bat, and won the match
gt_toss_bat_wins = df[(df['TossWinner'] == 'Gujarat Titans') &
                      (df['TossDecision'] == 'bat') &
                      (df['WinningTeam'] == 'Gujarat Titans')]

# Display the filtered matches
print("\nMatches where Gujarat Titans won the Toss, Elected to Bat, and won the match:")
print(gt_toss_bat_wins[['ID', 'Team1', 'Team2', 'City', 'Date']])
```

```
Matches where Gujarat Titans won the Toss, Elected to Bat, and won the match:
          ID           Team1                 Team2         City        Date
17  1304103  Gujarat Titans   Lucknow Super Giants         Pune  2022-05-10
39  1304081  Gujarat Titans  Kolkata Knight Riders  Navi Mumbai  2022-04-23
```

34. Use the file spotify.csv

- 1.Convert this file into a pandas Data Frame.
- 2.Display basic information like memory and data types for this data frame.
- 3.Display basic statistics like mean, std, quartiles, etc. for this data frame.
- 4.Create a correlation table for the data frame and comment about what kind of correlation is there between danceability and energy
- 5.Display first five rows for this data frame.
- 6.Display last five rows for this data frame.
- 7.Display the rows between 15 to 39 for this data frame.
- 8.Display the data only for last five rows and last five columns for this data frame.
- 9.Display the shape for this data frame.
- 10.Display the sum of NULL values for all the columns.
- 11.Remove first 3 columns from this Data Frame.
- 12.Remove first 10 rows from this Data Frame.
- 13.After removing first 3 columns and first 10 rows from this data frame find outliers for the column popularity.
- 14.After removing first 3 columns and first 10 rows from this data frame remove outliers for the column energy then display the data frame.
- 15.Display cross tabulation between time_signature and track_genre for actual Data Frame.

Unsupported Cell Type. Double-Click to inspect/edit the content.

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

df
```

	Unnamed: 0	track_id	artists	album_name	track_name	popularity	duration_ms	explicit	danceability	energy	...
0	0	5SuOikwiRyPMVoIQDJUgSV	Gen Hoshino	Comedy	Comedy	73	230666	False	0.676	0.4610	...
1	1	4qPNDBW1i3p13qLCt0Ki3A	Ben Woodward	Ghost (Acoustic)	Ghost - Acoustic	55	149610	False	0.420	0.1660	...
2	2	1iJBSr7s7jYXzM8EGcbK5b	Ingrid Michaelson;ZAYN	To Begin Again	To Begin Again	57	210826	False	0.438	0.3590	...
3	3	6lfxq3CG4xtTiEg7opyCyx	Kina Grannis	Crazy Rich Asians (Original Motion Picture Sou...	Can't Help Falling In Love	71	201933	False	0.266	0.0596	...
4	4	5vjLSffimiIP26QG5WcN2K	Chord Overstreet	Hold On	Hold On	82	198853	False	0.618	0.4430	...
...	...	...	...	...	...	...	...	...	...	...	...
113995	113995	2C3TZjDRiAzdyViavDJ217	Rainy Lullaby	#mindfulness - Soft Rain for Mindful Meditatio...	Sleep My Little Boy	21	384999	False	0.172	0.2350	...
113996	113996	1hIz5L4IB9hN3WRYPOCGPw	Rainy Lullaby	#mindfulness - Soft Rain for Mindful Meditatio...	Water Into Light	22	385000	False	0.174	0.1170	...
113997	113997	6x8ZfSoqDjuNa5SVP5QjvX	Cesária Evora	Best Of	Miss Perfumado	22	271466	False	0.629	0.3290	...
113998	113998	2e6sXL2bYv4bSz6VTdnfLs	Michael W. Smith	Change Your World	Friends	41	283893	False	0.587	0.5060	...
113999	113999	2hETkH7cOfqmz3LqZDHZf5	Cesária Evora	Miss Perfumado	Barbincor	22	241826	False	0.526	0.4870	...

114000 rows × 21 columns

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 114000 entries, 0 to 113999
Data columns (total 21 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   Unnamed: 0        114000 non-null  int64  
 1   track_id          114000 non-null  object 
 2   artists           113999 non-null  object 
 3   album_name        113999 non-null  object 
 4   track_name        113999 non-null  object 
 5   popularity        114000 non-null  int64  
 6   duration_ms       114000 non-null  int64  
 7   explicit          114000 non-null  bool   
 8   danceability      114000 non-null  float64
 9   energy            114000 non-null  float64
 10  key               114000 non-null  int64  
 11  loudness          114000 non-null  float64
 12  mode              114000 non-null  int64  
 13  speechiness       114000 non-null  float64
 14  acousticness      114000 non-null  float64
 15  instrumentalness  114000 non-null  float64
 16  liveness          114000 non-null  float64
 17  valence           114000 non-null  float64
 18  tempo             114000 non-null  float64
 19  time_signature    114000 non-null  int64  
 20  track_genre       114000 non-null  object 
dtypes: bool(1), float64(9), int64(6), object(5)
memory usage: 17.5+ MB
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df.describe()
```

	Unnamed: 0	popularity	duration_ms	danceability	energy	key	loudness	mode	speechiness	acoustici
count	114000.000000	114000.000000	1.140000e+05	114000.000000	114000.000000	114000.000000	114000.000000	114000.000000	114000.000000	114000.00(
mean	56999.500000	33.238535	2.280292e+05	0.566800	0.641383	5.309140	-8.258960	0.637553	0.084652	0.31
std	32909.109681	22.305078	1.072977e+05	0.173542	0.251529	3.559987	5.029337	0.480709	0.105732	0.33
min	0.000000	0.000000	0.000000e+00	0.000000	0.000000	0.000000	-49.531000	0.000000	0.000000	0.00(
25%	28499.750000	17.000000	1.740660e+05	0.456000	0.472000	2.000000	-10.013000	0.000000	0.035900	0.01(
50%	56999.500000	35.000000	2.129060e+05	0.580000	0.685000	5.000000	-7.004000	1.000000	0.048900	0.16
75%	85499.250000	50.000000	2.615060e+05	0.695000	0.854000	8.000000	-5.003000	1.000000	0.084500	0.59
max	113999.000000	100.000000	5.237295e+06	0.985000	1.000000	11.000000	4.532000	1.000000	0.965000	0.99

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
print("Weak positive correlation: Danceability increases as energy tends to increase.")
df.corr(numeric_only=True)
```

```
Weak positive correlation: Danceability increases as energy tends to increase.
```

	Unnamed: 0	popularity	duration_ms	explicit	danceability	energy	key	loudness	mode	speechiness	acousticness	ir
Unnamed: 0	1.000000	0.032142	-0.032743	-0.054736	0.003444	-0.055994	-0.005520	-0.027307	0.005107	-0.084952	0.076840	
popularity	0.032142	1.000000	-0.007101	0.044082	0.035448	0.001056	-0.003853	0.050423	-0.013931	-0.044927	-0.025472	
duration_ms	-0.032743	-0.007101	1.000000	-0.065263	-0.073426	0.058523	0.008114	-0.003470	-0.035556	-0.062600	-0.103788	
explicit	-0.054736	0.044082	-0.065263	1.000000	0.122507	0.096955	0.004484	0.108588	-0.037212	0.307952	-0.094403	
danceability	0.003444	0.035448	-0.073426	0.122507	1.000000	0.134325	0.036469	0.259077	-0.069219	0.108626	-0.171533	
energy	-0.055994	0.001056	0.058523	0.096955	0.134325	1.000000	0.048006	0.761690	-0.078362	0.142509	-0.733906	
key	-0.005520	-0.003853	0.008114	0.004484	0.036469	0.048006	1.000000	0.038590	-0.135916	0.020418	-0.040937	
loudness	-0.027307	0.050423	-0.003470	0.108588	0.259077	0.761690	0.038590	1.000000	-0.041764	0.060826	-0.589803	
mode	0.005107	-0.013931	-0.035556	-0.037212	-0.069219	-0.078362	-0.135916	-0.041764	1.000000	-0.046532	0.095553	
speechiness	-0.084952	-0.044927	-0.062600	0.307952	0.108626	0.142509	0.020418	0.060826	-0.046532	1.000000	-0.002186	
acousticness	0.076840	-0.025472	-0.103788	-0.094403	-0.171533	-0.733906	-0.040937	-0.589803	0.095553	-0.002186	1.000000	
instrumentalness	-0.070286	-0.095139	0.124371	-0.103404	-0.185606	-0.181879	-0.006823	-0.433477	-0.049955	-0.089616	0.104027	
liveness	0.033639	-0.005387	0.010321	0.032549	-0.131617	0.184796	-0.001600	0.076899	0.014012	0.205219	-0.020700	
valence	0.053111	-0.040534	-0.154479	-0.003381	0.477341	0.258934	0.034103	0.279848	0.021953	0.036635	-0.107070	
tempo	-0.025824	0.013205	0.024346	-0.002816	-0.050450	0.247851	0.010917	0.212446	0.000566	0.017273	-0.208224	
time_signature	-0.021115	0.031073	0.018225	0.038386	0.207218	0.187126	0.015065	0.191992	-0.024092	-0.000011	-0.176138	

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df.head()
```

	Unnamed: 0	track_id	artists	album_name	track_name	popularity	duration_ms	explicit	danceability	energy	...	loudn
0	0	5SuOikwiRyPMVoIQDJUgSV	Gen Hoshino	Comedy	Comedy	73	230666	False	0.676	0.4610	...	-6.
1	1	4qPNDBW1i3p13qLCt0Ki3A	Ben Woodward	Ghost (Acoustic)	Ghost - Acoustic	55	149610	False	0.420	0.1660	...	-17.
2	2	1iJBSr7s7jYXzM8EGcbK5b	Ingrid Michaelson;ZAYN	To Begin Again	To Begin Again	57	210826	False	0.438	0.3590	...	-9.
3	3	6lfxq3CG4xtTiEg7opyCyx	Kina Grannis	Crazy Rich Asians (Original Motion Picture Sou...	Can't Help Falling In Love	71	201933	False	0.266	0.0596	...	-18.
4	4	5vjLSffimiIP26QG5WcN2K	Chord Overstreet	Hold On	Hold On	82	198853	False	0.618	0.4430	...	-9.

5 rows × 21 columns

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df.tail()
```

	Unnamed: 0	track_id	artists	album_name	track_name	popularity	duration_ms	explicit	danceability	energy	...	loudne
113995	113995	2C3TZjDRiAzdyViavDJ217	Rainy Lullaby	#mindfulness - Soft Rain for Mindful Meditatio...	Sleep My Little Boy	21	384999	False	0.172	0.235	...	-16.3
113996	113996	1hIz5L4IB9hN3WRYPOCGPw	Rainy Lullaby	#mindfulness - Soft Rain for Mindful Meditatio...	Water Into Light	22	385000	False	0.174	0.117	...	-18.3
113997	113997	6x8ZfSoqDjuNa5SVP5QjvX	Cesária Evora	Best Of	Miss Perfumado	22	271466	False	0.629	0.329	...	-10.8
113998	113998	2e6sXL2bYv4bSz6VTdnfLs	Michael W. Smith	Change Your World	Friends	41	283893	False	0.587	0.506	...	-10.8
113999	113999	2hETkH7cOfqmz3LqZDHZf5	Cesária Evora	Miss Perfumado	Barbincor	22	241826	False	0.526	0.487	...	-10.2

5 rows × 21 columns

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df.tail()
```

	Unnamed: 0	track_id	artists	album_name	track_name	popularity	duration_ms	explicit	danceability	energy	...	loudne
113995	113995	2C3TZjDRiAzdyViavDJ217	Rainy Lullaby	#mindfulness - Soft Rain for Mindful Meditatio...	Sleep My Little Boy	21	384999	False	0.172	0.235	...	-16.3
113996	113996	1hIz5L4IB9hN3WRYPOCGPw	Rainy Lullaby	#mindfulness - Soft Rain for Mindful Meditatio...	Water Into Light	22	385000	False	0.174	0.117	...	-18.3
113997	113997	6x8ZfSoqDjuNa5SVP5QjvX	Cesária Evora	Best Of	Miss Perfumado	22	271466	False	0.629	0.329	...	-10.8
113998	113998	2e6sXL2bYv4bSz6VTdnfLs	Michael W. Smith	Change Your World	Friends	41	283893	False	0.587	0.506	...	-10.8
113999	113999	2hETkH7cOfqmz3LqZDHZf5	Cesária Evora	Miss Perfumado	Barbincor	22	241826	False	0.526	0.487	...	-10.2

5 rows × 21 columns

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df.head()+df.tail()
```

	Unnamed: 0	track_id	artists	album_name	track_name	popularity	duration_ms	explicit	danceability	energy	...	loudness	mode	speechi
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
113995	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
113996	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
113997	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
113998	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	
113999	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	

10 rows × 21 columns

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df.shape
```

```
(114000, 21)
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df.isna().sum()
```

```
Unnamed: 0          0
track_id            0
artists             1
album_name          1
track_name          1
popularity          0
duration_ms         0
explicit            0
danceability        0
energy              0
key                 0
loudness            0
mode                0
speechiness         0
acousticness        0
instrumentalness    0
liveness            0
valence             0
tempo               0
time_signature      0
track_genre         0
dtype: int64
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df.drop(df.columns[0:3],axis=1,inplace=True)
df.shape
```

```
(114000, 18)
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df.drop(range(0,10),axis=0,inplace=True)
df.shape
```

```
(113990, 18)
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
Q1 = df['popularity'].quantile(0.25)
Q3 = df['popularity'].quantile(0.75)
IQR = Q3 - Q1
# Determine the lower and upper bounds for outliers
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
# Identify outliers
outliers = df[(df['popularity'] < lower_bound) | (df['popularity'] > upper_bound)]
# Display the outliers
print("\nOutliers in the 'popularity' column:")
print(outliers)
```

```
Outliers in the 'popularity' column:
                    album_name                track_name  popularity  \
20001  Unholy (feat. Kim Petras)  Unholy (feat. Kim Petras)         100
81051  Unholy (feat. Kim Petras)  Unholy (feat. Kim Petras)         100

       duration_ms  explicit  danceability  energy  key  loudness  mode  \
20001       156943     False         0.714   0.472    2    -7.375     1
81051       156943     False         0.714   0.472    2    -7.375     1

       speechiness  acousticness  instrumentalness  liveness  valence  \
20001       0.0864         0.013          0.000005     0.266    0.238
81051       0.0864         0.013          0.000005     0.266    0.238

         tempo  time_signature track_genre
20001  131.121               4       dance
81051  131.121               4         pop
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df.shape
```

```
(113988, 18)
```

```
    Q1 = df['popularity'].quantile(0.25)
    Q3 = df['popularity'].quantile(0.75)
    IQR = Q3 - Q1

    # Determine the lower and upper bounds for outliers
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Identify outliers
    outliers = df[(df['popularity'] < lower_bound) | (df['popularity'] > upper_bound)]

    # Remove outliers
    df = df[~((df['popularity'] < lower_bound) | (df['popularity'] > upper_bound))]
```

```
df.shape
```

```
(113988, 18)
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
pd.crosstab(df.time_signature,df.track_genre)
```

track_genre	acoustic	afrobeat	alt-rock	alternative	ambient	anime	black-metal	bluegrass	blues	brazil	...	spanish	study	swedish	synth-pop	tang
time_signature																
0	0	0	1	0	3	0	0	0	0	0	...	0	0	0	0	
1	9	5	1	2	39	7	20	5	6	5	...	2	6	6	2	1
3	99	52	56	64	264	63	129	80	97	42	...	55	49	49	16	16
4	869	930	940	922	648	923	838	909	894	939	...	937	934	928	981	78
5	13	13	2	12	46	7	13	6	3	14	...	6	11	17	1	3

5 rows × 114 columns

35

1. Load the dataset into a pandas DataFrame (data_result.csv) and answer the following questions.
2. View the first few rows of the dataset
3. Check the shape of the dataset
4. View the first last rows of the dataset
5. Get summary statistics of numerical columns
6. Get summary statistics of numerical columns with 0.58 and 0.87 percentiles
7. Get summary statistics of all types of columns
8. Information of all columns
9. Check for missing values
10. Removing duplicates if duplicates
11. List out female students who have greater than 7 spi in all semesters.
12. Find number of students those who have greater than 8 spi in all 5 semesters.
13. Find outliers of sem 4 result. Also represent statistical analysis with visualization.(boxplot)

Unsupported Cell Type. Double-Click to inspect/edit the content.

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

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df.head()
```

	1st	2nd	3rd	4th	5th	College Code	Gender	Roll no.	Subject Code
0	8.11	7.68	7.11	7.43	8.18	115	Female	17020	16
1	6.48	5.90	4.15	4.29	4.96	115	Male	17021	16
2	8.41	8.24	7.52	8.25	7.75	115	Female	17022	16
3	7.33	6.83	6.33	6.79	6.89	115	Male	17023	16
4	7.89	7.34	7.22	7.32	7.46	115	Male	17024	16

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df.tail()
```

	1st	2nd	3rd	4th	5th	College Code	Gender	Roll no.	Subject Code
41	6.30	6.24	5.85	6.36	7.00	115	Male	17061	16
42	7.78	6.93	7.44	7.86	8.21	115	Male	17062	16
43	8.22	6.66	7.07	7.29	7.61	115	Male	17063	16
44	7.67	7.07	7.04	7.07	6.93	115	Male	17064	16
45	8.41	7.59	7.41	7.89	8.11	115	Male	17065	16

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df.shape
```

```
(46, 9)
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df.tail()
```

	1st	2nd	3rd	4th	5th	College Code	Gender	Roll no.	Subject Code
41	6.30	6.24	5.85	6.36	7.00	115	Male	17061	16
42	7.78	6.93	7.44	7.86	8.21	115	Male	17062	16
43	8.22	6.66	7.07	7.29	7.61	115	Male	17063	16
44	7.67	7.07	7.04	7.07	6.93	115	Male	17064	16
45	8.41	7.59	7.41	7.89	8.11	115	Male	17065	16

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df.describe()
```

	1st	2nd	3rd	4th	5th	College Code	Roll no.	Subject Code
count	46.000000	46.000000	46.000000	46.000000	46.000000	46.0	46.000000	46.0
mean	7.397609	6.930217	6.703043	7.237826	7.527609	115.0	17042.500000	16.0
std	0.798391	0.910425	0.917324	1.057981	0.967963	0.0	13.422618	0.0
min	5.670000	4.280000	4.150000	4.290000	4.860000	115.0	17020.000000	16.0
25%	6.787500	6.350000	6.217500	6.650000	6.890000	115.0	17031.250000	16.0
50%	7.440000	6.810000	7.000000	7.290000	7.625000	115.0	17042.500000	16.0
75%	8.040000	7.590000	7.322500	7.890000	8.210000	115.0	17053.750000	16.0
max	8.890000	8.720000	8.370000	9.250000	9.000000	115.0	17065.000000	16.0

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df.describe(percentiles=[0.58,0.87])
```

	1st	2nd	3rd	4th	5th	College Code	Roll no.	Subject Code
count	46.000000	46.000000	46.000000	46.000000	46.000000	46.0	46.000000	46.0
mean	7.397609	6.930217	6.703043	7.237826	7.527609	115.0	17042.500000	16.0
std	0.798391	0.910425	0.917324	1.057981	0.967963	0.0	13.422618	0.0
min	5.670000	4.280000	4.150000	4.290000	4.860000	115.0	17020.000000	16.0
50%	7.440000	6.810000	7.000000	7.290000	7.625000	115.0	17042.500000	16.0
58%	7.787000	6.944000	7.114000	7.647000	7.860000	115.0	17046.100000	16.0
87%	8.220000	8.000000	7.530500	8.336500	8.531500	115.0	17059.150000	16.0
max	8.890000	8.720000	8.370000	9.250000	9.000000	115.0	17065.000000	16.0

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df.describe(include="all")
```

	1st	2nd	3rd	4th	5th	College Code	Gender	Roll no.	Subject Code
count	46.000000	46.000000	46.000000	46.000000	46.000000	46.0	46	46.000000	46.0
unique	NaN	NaN	NaN	NaN	NaN	NaN	2	NaN	NaN
top	NaN	NaN	NaN	NaN	NaN	NaN	Male	NaN	NaN
freq	NaN	NaN	NaN	NaN	NaN	NaN	38	NaN	NaN
mean	7.397609	6.930217	6.703043	7.237826	7.527609	115.0	NaN	17042.500000	16.0
std	0.798391	0.910425	0.917324	1.057981	0.967963	0.0	NaN	13.422618	0.0
min	5.670000	4.280000	4.150000	4.290000	4.860000	115.0	NaN	17020.000000	16.0
25%	6.787500	6.350000	6.217500	6.650000	6.890000	115.0	NaN	17031.250000	16.0
50%	7.440000	6.810000	7.000000	7.290000	7.625000	115.0	NaN	17042.500000	16.0
75%	8.040000	7.590000	7.322500	7.890000	8.210000	115.0	NaN	17053.750000	16.0
max	8.890000	8.720000	8.370000	9.250000	9.000000	115.0	NaN	17065.000000	16.0

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 46 entries, 0 to 45
Data columns (total 9 columns):
```

```
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   1st           46 non-null     float64
 1   2nd           46 non-null     float64
 2   3rd           46 non-null     float64
 3   4th           46 non-null     float64
 4   5th           46 non-null     float64
 5   College Code  46 non-null     int64
 6   Gender        46 non-null     object
 7   Roll no.      46 non-null     int64
 8   Subject Code  46 non-null     int64
dtypes: float64(5), int64(3), object(1)
memory usage: 3.4+ KB
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df.isnull().sum()
```

```
1st             0
2nd             0
3rd             0
4th             0
5th             0
College Code    0
Gender          0
Roll no.        0
Subject Code    0
dtype: int64
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df.drop_duplicates(inplace=True)
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df[(df["Gender"]=="Female")&(df["1st"]>7)&(df["2nd"]>7)&(df["3rd"]>7)&(df["4th"]>7)&(df["5th"]>7)]
```

	1st	2nd	3rd	4th	5th	College Code	Gender	Roll no.	Subject Code
0	8.11	7.68	7.11	7.43	8.18	115	Female	17020	16
2	8.41	8.24	7.52	8.25	7.75	115	Female	17022	16
21	8.33	8.72	7.81	8.04	8.93	115	Female	17041	16
31	8.89	8.31	7.30	9.25	8.50	115	Female	17051	16

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df[(df["1st"]>8)&(df["2nd"]>8)&(df["3rd"]>8)&(df["4th"]>8)&(df["5th"]>8)].shape[0]
```

```
1
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
target_column = df.columns[3]  # Access the 4th column

# Calculate quartiles for IQR outlier detection
Q1 = df[target_column].quantile(0.25)
Q3 = df[target_column].quantile(0.75)
IQR = Q3 - Q1

# Define outlier boundaries based on IQR and threshold (1.5)
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Filter DataFrame using boolean indexing and query
df_filtered = df[(df[target_column] < lower_bound) | (df[target_column] > upper_bound)]

print(df.shape)
print(df_filtered.shape)
```

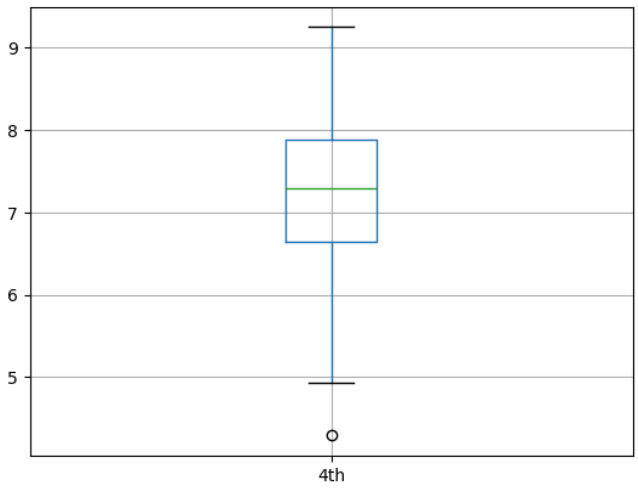
```
(46, 9)
(1, 9)
```

```
df.describe()
```

	1st	2nd	3rd	4th	5th	College Code	Roll no.	Subject Code
count	46.000000	46.000000	46.000000	46.000000	46.000000	46.0	46.000000	46.0
mean	7.397609	6.930217	6.703043	7.237826	7.527609	115.0	17042.500000	16.0
std	0.798391	0.910425	0.917324	1.057981	0.967963	0.0	13.422618	0.0
min	5.670000	4.280000	4.150000	4.290000	4.860000	115.0	17020.000000	16.0
25%	6.787500	6.350000	6.217500	6.650000	6.890000	115.0	17031.250000	16.0
50%	7.440000	6.810000	7.000000	7.290000	7.625000	115.0	17042.500000	16.0
75%	8.040000	7.590000	7.322500	7.890000	8.210000	115.0	17053.750000	16.0
max	8.890000	8.720000	8.370000	9.250000	9.000000	115.0	17065.000000	16.0

```
df.boxplot("4th")
```

```
<Axes: >
```



"Use the file movies.csv which contains 1629 rows and 18 columns. Read this csv file and display the basic information like memory and data types for this data frame.

Write python code for the following cases:

1. List out Movies Released in Year 2019.
2. How Many Movies are having IMDB Rating > 7 (Display Number of Movies).
3. List out the Movies with 'title' and 'story' whose IMDB Votes > 20000.
4. List out Movies Released in Year 2018, Display only Movie Title with Release Date of Year 2018 Movies.
5. Display only Movie Title with its Wikipedia Link."

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df.columns
```

```
Index(['title_x', 'imdb_id', 'poster_path', 'wiki_link', 'title_y',
       'original_title', 'is_adult', 'year_of_release', 'runtime', 'genres',
       'imdb_rating', 'imdb_votes', 'story', 'summary', 'tagline', 'actors',
       'wins_nominations', 'release_date'],
      dtype='object')
```

```
import pandas as pd

# Load the dataset into a DataFrame
df = pd.read_csv("movies.csv")
# 1. List out Movies Released in Year 2019
# 1. List out Movies Released in Year 2019
movies_2019 = df[df['year_of_release'] == 2019]
print("\nMovies Released in Year 2019:")
print(movies_2019.shape[0])
print(movies_2019)
```

```
Movies Released in Year 2019:
75
                                  title_x     imdb_id  \
0                Uri: The Surgical Strike   tt8291224
1                            Battalion 609   tt9472208
2     The Accidental Prime Minister (film)   tt6986710
3                          Why Cheat India   tt8108208
```

```
6                          Fraud Saiyaan   tt5013008
...                                  ...         ...
76                     Commando 3 (film)   tt8983168
77                            Mardaani 2   tt5668770
78                             Dabangg 3   tt7059844
79                            Good Newwz   tt8504014
1627                               Daaka  tt10833860

                                      poster_path  \
0     https://upload.wikimedia.org/wikipedia/en/thum...
1                                               NaN
2     https://upload.wikimedia.org/wikipedia/en/thum...
3     https://upload.wikimedia.org/wikipedia/en/thum...
6     https://upload.wikimedia.org/wikipedia/en/thum...
...                                                ...
76    https://upload.wikimedia.org/wikipedia/en/thum...
77    https://upload.wikimedia.org/wikipedia/en/thum...
78    https://upload.wikimedia.org/wikipedia/en/thum...
79                                              NaN
1627  https://upload.wikimedia.org/wikipedia/en/thum...

                                        wiki_link  \
0     https://en.wikipedia.org/wiki/Uri:_The_Surgica...
1          https://en.wikipedia.org/wiki/Battalion_609
2     https://en.wikipedia.org/wiki/The_Accidental_P...
3         https://en.wikipedia.org/wiki/Why_Cheat_India
6           https://en.wikipedia.org/wiki/Fraud_Saiyaan
...                                                 ...
76      https://en.wikipedia.org/wiki/Commando_3_(film)
77             https://en.wikipedia.org/wiki/Mardaani_2
78              https://en.wikipedia.org/wiki/Dabangg_3
79             https://en.wikipedia.org/wiki/Good_Newwz
1627                https://en.wikipedia.org/wiki/Daaka

                           title_y                 original_title  is_adult  \
0          Uri: The Surgical Strike       Uri: The Surgical Strike         0
1                       Battalion 609                  Battalion 609         0
2     The Accidental Prime Minister  The Accidental Prime Minister         0
3                   Why Cheat India                Why Cheat India         0
6                     Fraud Saiyaan                  Fraud Saiyyan         0
...                             ...                            ...       ...
76                      Commando 3                     Commando 3         0
77                      Mardaani 2                     Mardaani 2         0
78                       Dabangg 3                      Dabangg 3         0
79                      Good Newwz                     Good Newwz         0
1627                         Daaka                          Daaka         0

      year_of_release runtime                genres  imdb_rating  imdb_votes  \
0                2019     138       Action|Drama|War          8.4       35112
1                2019     131                   War          4.1          73
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
# 2. How Many Movies are having IMDB Rating > 7 (Display Number of Movies)
high_rating_movies = df[df['imdb_rating'] > 7]
high_rating_movies_count = high_rating_movies.shape[0]
print(f"\nNumber of Movies with IMDB Rating > 7: {high_rating_movies_count}")
```

```
Number of Movies with IMDB Rating > 7: 331
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
# 3. List out the Movies with 'title_x' and 'story' whose IMDB Votes > 20000
popular_movies = df[df['imdb_votes'] > 20000][['title_x', 'story']]
print("\nMovies with IMDB Votes > 20000 (Title and Story):")
print(popular_movies)
```

```
Movies with IMDB Votes > 20000 (Title and Story):
                        title_x  \
0       Uri: The Surgical Strike
11                    Gully Boy
36                   Kabir Singh
74                   Dil Bechara
88                     Padmaavat
...                          ...
1490     Devdas (2002 Hindi film)
1565  Kabhi Khushi Kabhie Gham...
1567                      Lagaan
1568                      Lagaan
1571               Dil Chahta Hai

                                                  story
0     Divided over five chapters  the film chronicle...
11    Gully Boy is a film about a 22-year-old boy "M...
36    This Sandeep Vanga directorial is a remake of ...
74              A love story about two cancer patients.
88    This fictional story is set in 13th century me...
...                                                 ...
1490  Devdas Mukherji is black-listed by his multi-m...
1565  Yashvardhan Raichand lives a very wealthy life...
1567  This is the story about the resilience shown b...
1568  This is the story about the resilience shown b...
1571  Three young men Akash  Sameer  and Siddharth a...
```

```
[105 rows x 2 columns]
```

4. List out Movies Released in Year 2018, Display only Movie Title with Release Date of Year 2018 Movies.

```
#  List out Movies Released in Year 2018, Display only Movie Title with Release Date of Year 2018 Movies
movies_2018 = df[df['year_of_release'] == 2018][['title_x', 'release_date']]
print("\nMovies Released in Year 2018 (Title and Release Date):")
print(movies_2018)
```

```
Movies Released in Year 2018 (Title and Release Date):
                        title_x              release_date
4                Evening Shadows   11 January 2019 (India)
5                    Soni (film)     18 January 2019 (USA)
16       Mard Ko Dard Nahi Hota       22 March 2019 (USA)
17                  Hamid (film)     15 March 2019 (India)
20    Mere Pyare Prime Minister     15 March 2019 (India)
...                          ...                      ...
156                 Rajma Chawal  30 November 2018 (India)
157            Zero (2018 film)    21 December 2018 (USA)
158                      Simmba    28 December 2018 (USA)
166          Thugs of Hindostan     8 November 2018 (USA)
1626             Sabse Bada Sukh                      NaN

[79 rows x 2 columns]
```

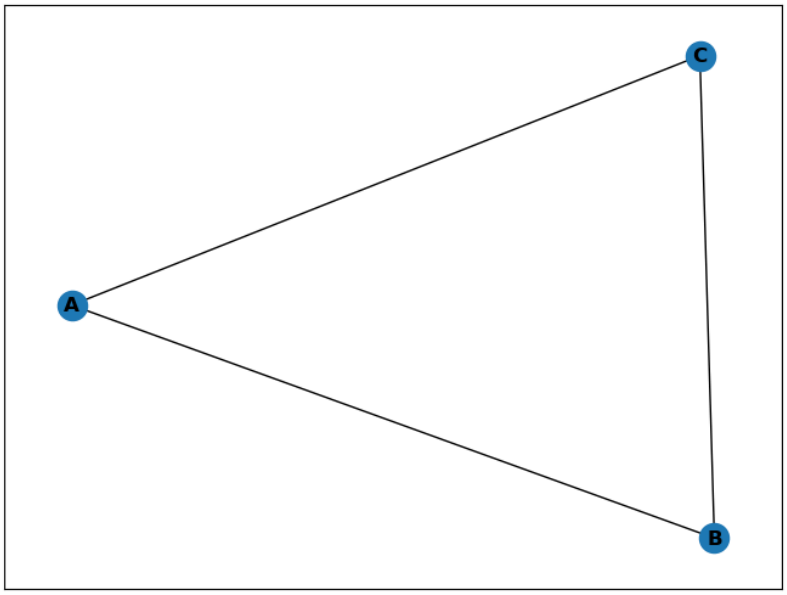
5. Display only Movie Title with its Wikipedia Link."

```
# 5. Display only Movie Title with its Wikipedia Link
movie_wikipedia_links = df[['title_x', 'wiki_link']]
print("\nMovie Titles with Wikipedia Links:")
print(movie_wikipedia_links)
```

```
Movie Titles with Wikipedia Links:
                                title_x  \
0                 Uri: The Surgical Strike
1                          Battalion 609
2     The Accidental Prime Minister (film)
3                          Why Cheat India
4                          Evening Shadows
...                                    ...
1624                 Tera Mera Saath Rahen
1625                  Yeh Zindagi Ka Safar
1626                       Sabse Bada Sukh
1627                                Daaka
1628                              Humsafar

                                      wiki_link
0     https://en.wikipedia.org/wiki/Uri:_The_Surgica...
1           https://en.wikipedia.org/wiki/Battalion_609
2     https://en.wikipedia.org/wiki/The_Accidental_P...
3         https://en.wikipedia.org/wiki/Why_Cheat_India
4         https://en.wikipedia.org/wiki/Evening_Shadows
...                                                ...
1624  https://en.wikipedia.org/wiki/Tera_Mera_Saath_...
1625  https://en.wikipedia.org/wiki/Yeh_Zindagi_Ka_S...
1626      https://en.wikipedia.org/wiki/Sabse_Bada_Sukh
1627                https://en.wikipedia.org/wiki/Daaka
1628             https://en.wikipedia.org/wiki/Humsafar

[1629 rows x 2 columns]
```

70 Write a python program which creates following graph using networkx module in python

```
import matplotlib.pyplot as plt
import networkx as nx

# Create a graph object
G = nx.Graph()

# Add nodes with labels to the graph
G.add_nodes_from([("A", {"label": "Node A"}), ("B", {"label": "Node B"}), ("C", {"label": "Node C"})])

# Add edges between the nodes
G.add_edges_from([("A", "B"), ("B", "C"), ("C", "A")])

# Draw the graph with a spring layout and node labels  # Get node positions
nx.draw_networkx(G, with_labels=True, font_weight='bold')  # Draw with labels

# Display the graph
plt.show()
```

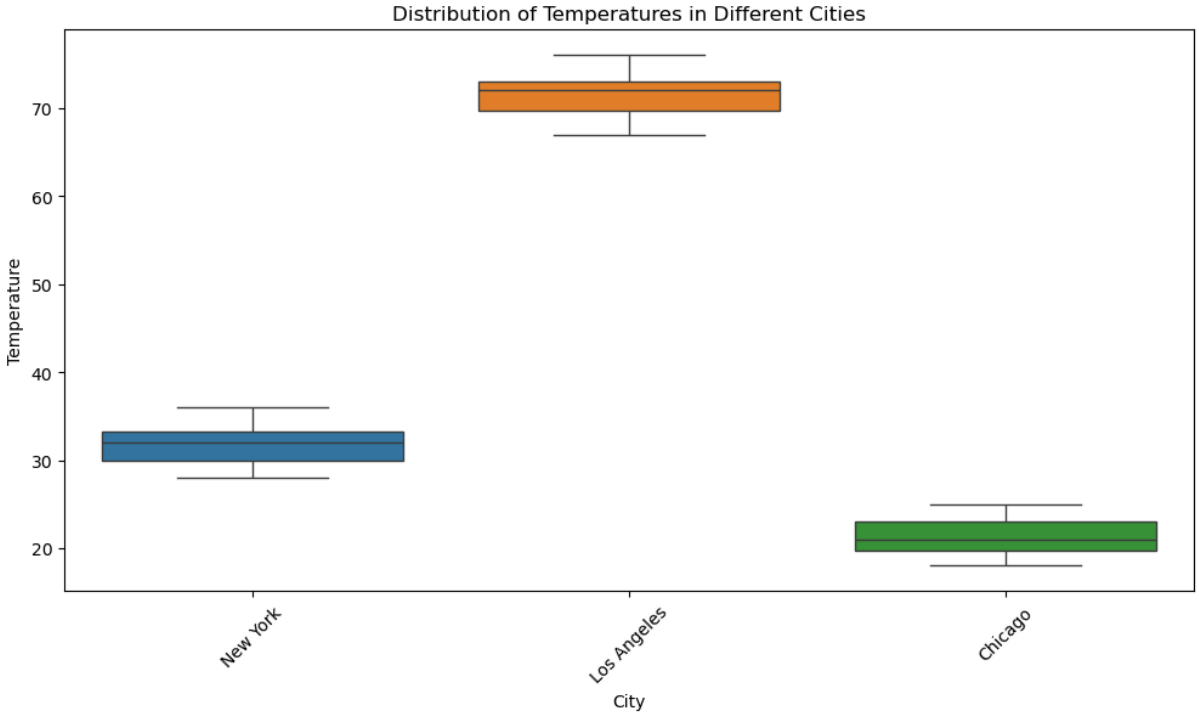


71 Create a boxplot of the distribution of temperatures in different cities. Take data from 'temperatures.csv' from below:

https://raw.githubusercontent.com/kavit88/Data-Sets/main/temperatures.csv

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset into a DataFrame
url = 'https://raw.githubusercontent.com/kavit88/Data-Sets/main/temperatures.csv'
df = pd.read_csv(url)

# Display the first few rows of the DataFrame to understand its structure
print("\nFirst few rows of the DataFrame:")
print(df.head())

# Create a boxplot of the distribution of temperatures in different cities
plt.figure(figsize=(12, 6))
sns.boxplot(data=df[["New York" , "Los Angeles" , "Chicago"]])
plt.title('Distribution of Temperatures in Different Cities')
plt.xlabel('City')
plt.ylabel('Temperature')
plt.xticks(rotation=45)
plt.show()
```

```
First few rows of the DataFrame:
   New York  Los Angeles  Chicago        Date
0        30           70       20  01-05-2023
1        35           72       22  02-05-2023
2        28           68       18  03-05-2023
3        32           75       24  04-05-2023
4        33           73       21  05-05-2023
<Figure size 1200x600 with 0 Axes>
```

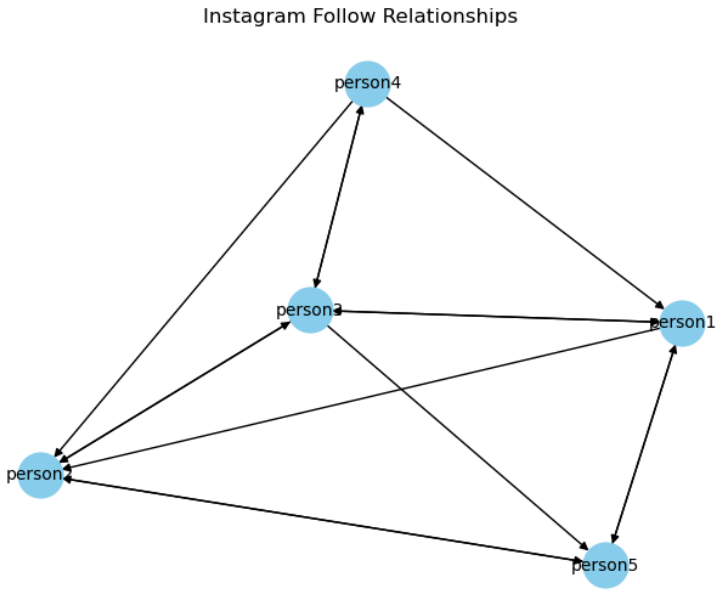


72. The following dictionary shows how five people follow each other on Instagram:

instagram = {'person1': [0,1,1,0,1], 'person2': [0,0,1,0,1], 'person3': [1,1,0,1,1],'person4': [1,1,1,0,0], 'person5': [1,1,0,0,0]} E.g., the list for person1 has the value on index 2 as 1 which means person1 followsperson3 and a directed edge should be added from person1 to person3.

Using networkx library, create a directed graph.

```
import networkx as nx
import matplotlib.pyplot as plt

# Define the follow relationships
instagram = {
    'person1': [0, 1, 1, 0, 1],
    'person2': [0, 0, 1, 0, 1],
    'person3': [1, 1, 0, 1, 1],
    'person4': [1, 1, 1, 0, 0],
    'person5': [1, 1, 0, 0, 0]
}

# Create a directed graph
G = nx.DiGraph()

# Add nodes
for person in instagram.keys():
    G.add_node(person)

# Add edges
for follower, follows in instagram.items():
    for i, follow in enumerate(follows):
        if follow == 1:
            G.add_edge(follower, f'person{i+1}')

# Visualize the graph
nx.draw(G,with_labels=True, node_size=700, node_color='skyblue', font_size=10, arrows=True)
plt.title('Instagram Follow Relationships')
plt.show()
```

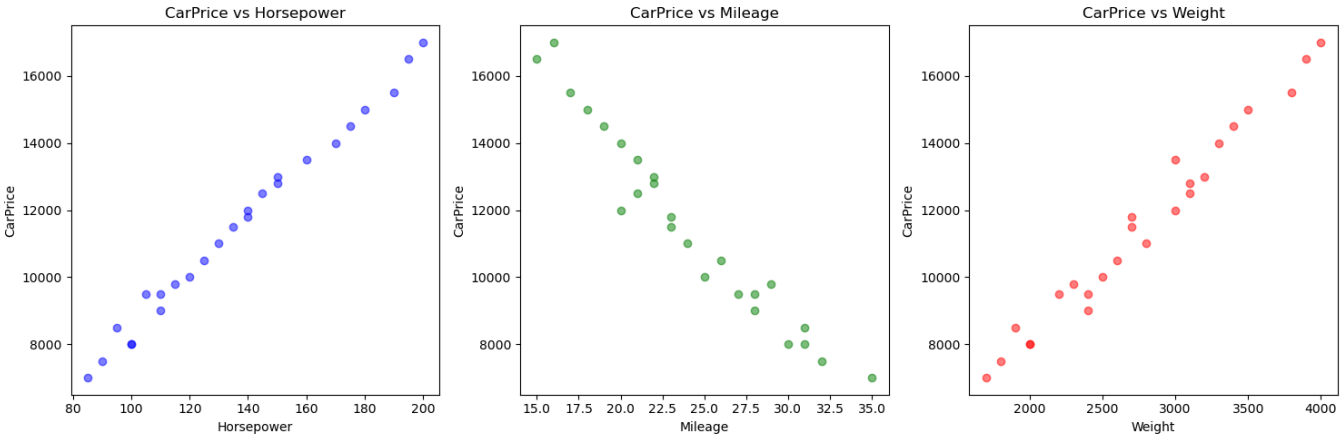


73 You have been given a dataset of car prices and their respective horsepower, mileage, and weight. You have been tasked to analyze the relationship between these variables and create a scatter plot to visualize the patterns.

Dataset: The dataset, named "car_data.csv" :

https://raw.githubusercontent.com/kavit88/Data-Sets/main/car_data.csv

```
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset into a DataFrame
url = 'https://raw.githubusercontent.com/kavit88/Data-Sets/main/car_data.csv'
df = pd.read_csv(url)



# Create scatter plots
plt.figure(figsize=(15, 5))

# Scatter plot between CarPrice and Horsepower
plt.subplot(1, 3, 1)
plt.scatter(df['Horsepower'], df['Price'], color='blue', alpha=0.5)
plt.title('CarPrice vs Horsepower')
plt.xlabel('Horsepower')
plt.ylabel('CarPrice')

# Scatter plot between CarPrice and Mileage
plt.subplot(1, 3, 2)
plt.scatter(df['Mileage'], df['Price'], color='green', alpha=0.5)
plt.title('CarPrice vs Mileage')
plt.xlabel('Mileage')
plt.ylabel('CarPrice')

# Scatter plot between CarPrice and Weight
plt.subplot(1, 3, 3)
plt.scatter(df['Weight'], df['Price'], color='red', alpha=0.5)
plt.title('CarPrice vs Weight')
plt.xlabel('Weight')
plt.ylabel('CarPrice')

plt.tight_layout()
plt.show()
```

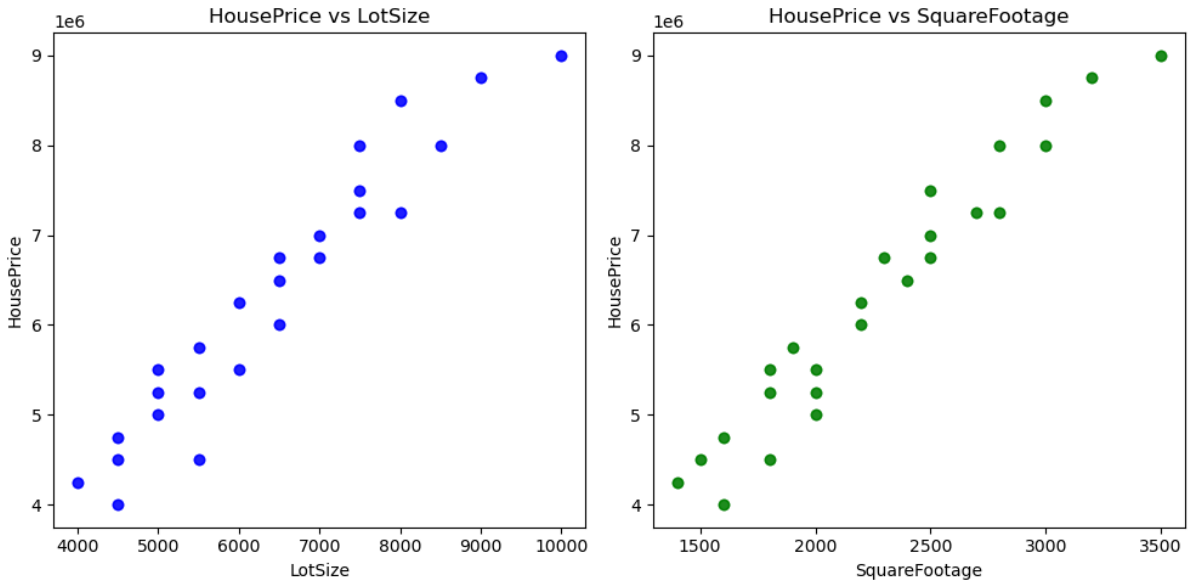


74 You have been given a dataset of house prices and their respective lot size and square footage. Your task is to create a scatter plot to determine if there is any correlation between these variables.

Dataset: The dataset, named "house_data.csv":

https://raw.githubusercontent.com/kavit88/Data-Sets/main/house_data.csv

```
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset into a DataFrame
url = 'https://raw.githubusercontent.com/kavit88/Data-Sets/main/house_data.csv'
df = pd.read_csv(url)


# Create scatter plots

# Scatter plot between HousePrice and LotSize
plt.subplot(1, 2, 1)
plt.scatter(df['LotSize'], df['Price'], color='blue', alpha=0.5)
plt.title('HousePrice vs LotSize')
plt.xlabel('LotSize')
plt.ylabel('HousePrice')

# Scatter plot between HousePrice and SquareFootage
plt.subplot(1, 2, 2)
plt.scatter(df['SqFt'], df['Price'], color='green', alpha=0.5)
plt.title('HousePrice vs SquareFootage')
plt.xlabel('SquareFootage')
plt.ylabel('HousePrice')
plt.tight_layout()
plt.show()
```



75 Use the file heights_weights.csv which contains 10000 non-null values for heights and weights. The Male column shows 1 if the person is a Male and 0 if the person is a Female. Take file of dataset from:

https://raw.githubusercontent.com/kavit88/Data-Sets/main/heights_weights.csv

1. Convert this file into a pandas Data Frame.
2. Display basic information like memory and data types for this data frame.
3. Display basic statistics like mean, std, quartiles, etc. for this data frame.
4. Create a correlation table for the data frame and comment about what kind ofcorrelation is there between Height and Weight.
5. Do Height and Weight contain any outliers? Answer by creating boxplots for both.
6. Finally, create a scatter plot of Weight v/s Height with the following specifications: (i) use + sign, colour green and size 50 for markers. (ii) Label X Axis as Weight and Y Axis as Height. (iii) Display title on top as Weight vs Height

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
import pandas as pd

# Load the dataset into a DataFrame
url = 'https://raw.githubusercontent.com/kavit88/Data-Sets/main/heights_weights.csv'
df = pd.read_csv(url)
# Display the first few rows of the DataFrame
print("\nFirst few rows of the DataFrame:")
print(df.head())
```

```
First few rows of the DataFrame:
      Height      Weight  Male
0  73.847017  241.893563     1
1  68.781904  162.310473     1
2  74.110105  212.740856     1
3  71.730978  220.042470     1
4  69.881796  206.349801     1
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Height  10000 non-null  float64
 1   Weight  10000 non-null  float64
 2   Male    10000 non-null  int64
dtypes: float64(2), int64(1)
memory usage: 234.5 KB
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df.describe()
```

	Height	Weight	Male
count	10000.000000	10000.000000	10000.000000
mean	66.367560	161.440357	0.500000
std	3.847528	32.108439	0.500025
min	54.263133	64.700127	0.000000
25%	63.505620	135.818051	0.000000
50%	66.318070	161.212928	0.500000
75%	69.174262	187.169525	1.000000
max	78.998742	269.989698	1.000000

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df.corr(numeric_only=True)
```

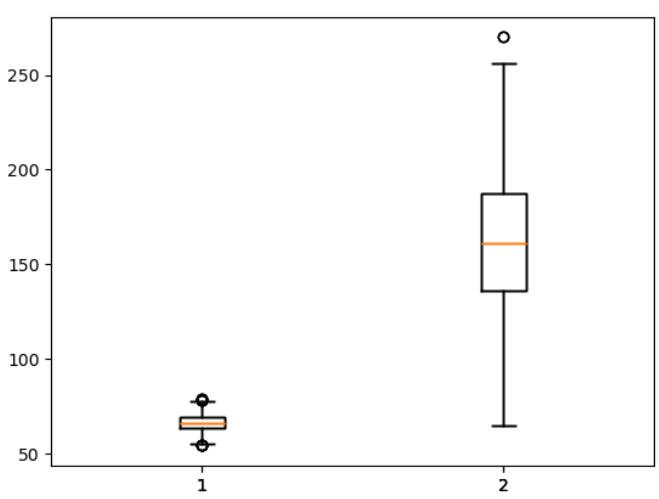
	Height	Weight	Male
Height	1.000000	0.924756	0.691072
Weight	0.924756	1.000000	0.796723
Male	0.691072	0.796723	1.000000

```
print("Strong relation:posative high vale")
```

```
Strong relation:posative high vale
```

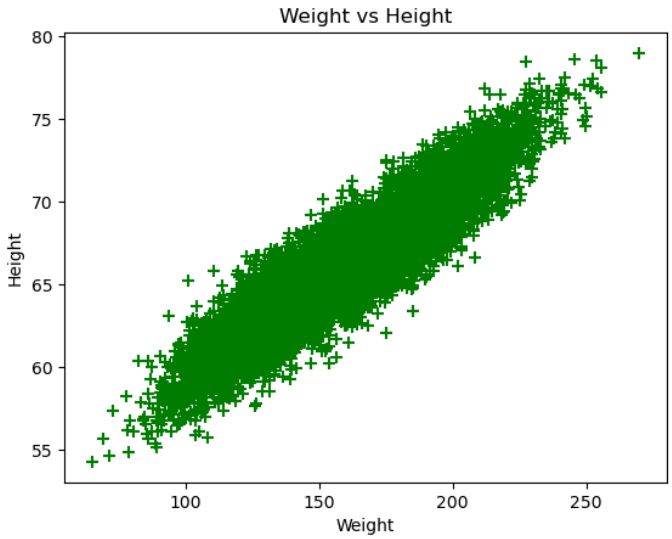
Unsupported Cell Type. Double-Click to inspect/edit the content.

```
plt.boxplot(df[["Height","Weight"]])
plt.show()
```



Unsupported Cell Type. Double-Click to inspect/edit the content.

```
plt.scatter(df['Weight'], df['Height'], marker='+', color='green', s=50)
plt.title('Weight vs Height')
plt.xlabel('Weight')
plt.ylabel('Height')
plt.show()
```

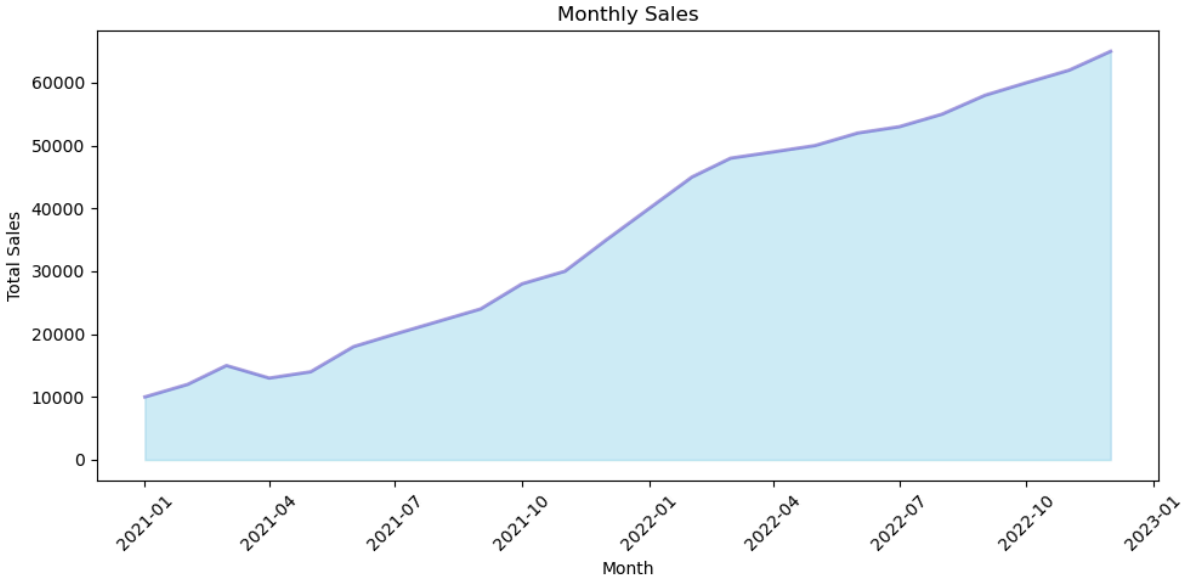


76 The file "sales.csv" contains the monthly sales data for a store over a year. Each row contains the month (in the format "yyyy-mm"), the total sales for that month, and the number of items sold. Create a pandas DataFrame from this data and plot the monthly sales using an area plot. Take the dataset from below:

https://raw.githubusercontent.com/kavit88/Data-Sets/main/sales.csv

```
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset into a DataFrame
url = 'https://raw.githubusercontent.com/kavit88/Data-Sets/main/sales.csv'
df = pd.read_csv(url)

## Convert the month column to datetime format
df['Month'] = pd.to_datetime(df['Month'])

# Plot the monthly sales using an area plot
plt.figure(figsize=(10, 5))
plt.fill_between(df['Month'], df['Total Sales'], color="skyblue", alpha=0.4)
plt.plot(df['Month'], df['Total Sales'], color="Slateblue", alpha=0.6, linewidth=2)
plt.title('Monthly Sales')
plt.xlabel('Month')
plt.ylabel('Total Sales')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
<Figure size 1000x500 with 0 Axes>
<Figure size 1000x500 with 0 Axes>
```



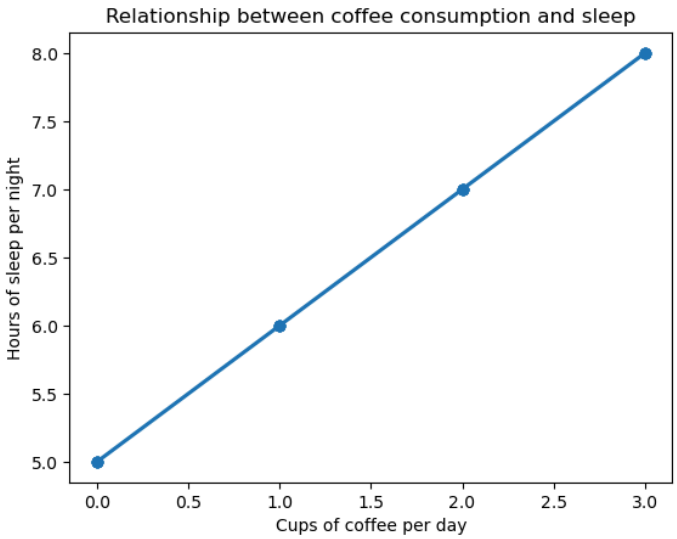
77 The file "survey.csv" contains the results of a survey that asks people how many hours they sleep per night, how much coffee they drink per day, and how many hours they spend exercising per week. Create a pandas DataFrame from this data and plot the relationships between these variables using regression plots. Specifically, create the following plots:

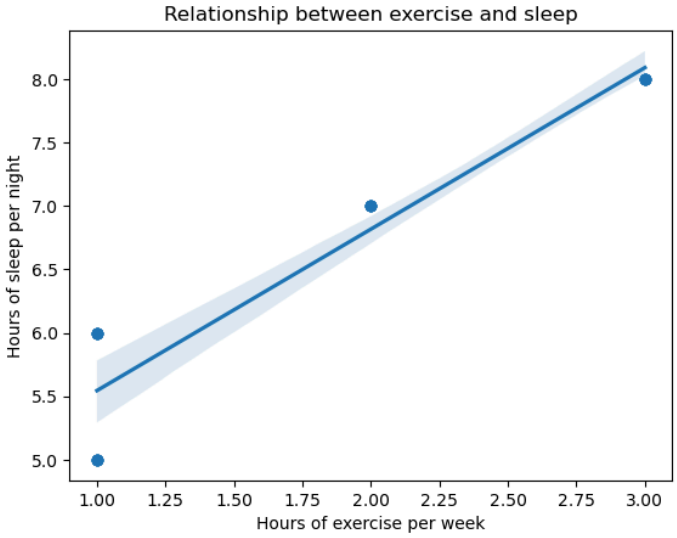
1. A regression plot of hours of sleep versus cups of coffee per day, with a regression line and confidence interval.
2. A regression plot of hours of sleep versus hours of exercise per week, with a regression line and confidence interval.
3. A regression plot of cups of coffee per day versus hours of exercise per week, with a regression line and confidence interval.

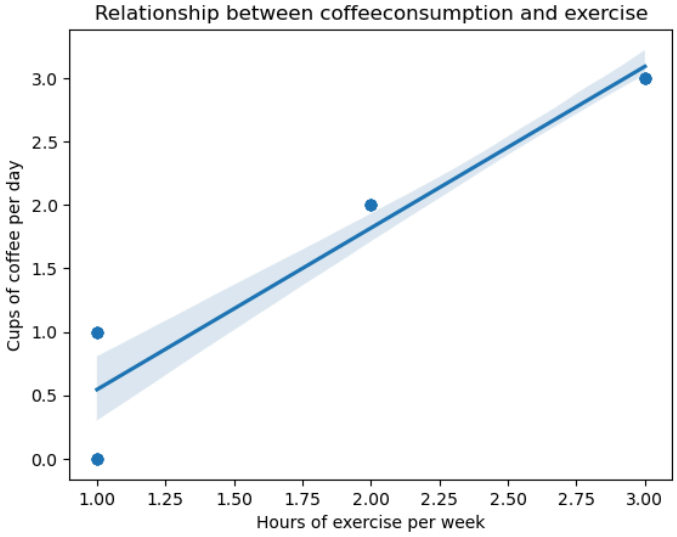
Label each axis appropriately and give each plot a title. Take Dataset from below: https://raw.githubusercontent.com/kavit88/Data-Sets/main/survey.csv

```
import pandas as pd
import seaborn as sns
# Load the data into a pandas DataFrame
survey_df = pd.read_csv("https://raw.githubusercontent.com/kavit88/Data-Sets/main/survey.csv")
# Plot regression of hours of sleep
#versus cups of coffee per day
sns.regplot(x="cups_of_coffee_per_day",
y="hours_of_sleep", data=survey_df)
plt.xlabel("Cups of coffee per day")
plt.ylabel("Hours of sleep per night")
plt.title("Relationship between coffee consumption and sleep")
plt.show()
# Plot regression of hours of sleep
#versus hours of exercise per week
sns.regplot(x="hours_of_exercise_per_week",
y="hours_of_sleep", data=survey_df)
plt.xlabel("Hours of exercise per week")
plt.ylabel("Hours of sleep per night")
plt.title("Relationship between exercise and sleep")
plt.show()
# Plot regression of cups of coffee per day
#versus hours of exercise per week
sns.regplot(x="hours_of_exercise_per_week",
y="cups_of_coffee_per_day", data=survey_df)
plt.xlabel("Hours of exercise per week")
plt.ylabel("Cups of coffee per day")
plt.title("Relationship between coffeeconsumption and exercise")
plt.show()
```

{x}





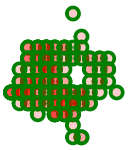


78 Use the California_Houses.csv file to create a map with the first 200 rows using the latitudes and longitudes given in the file with the following customizations:

1. Colour of circle markers should be green with red fill and the type of map should be stamen terrain
2. Add pop up labels using the population from the file. Take the dataset fom below: https://raw.githubusercontent.com/kavit88/Data-Sets/main/California_Houses.csv

```
import folium
import pandas as pd
# Load the data from the CSV file into a pandas DataFrame
data = pd.read_csv('California_Houses.csv')
# Create a map centered at the mean latitude
# and longitude of the first 200 rows
m = folium.Map(location=[data.iloc[:200]['latitude'].mean(), data.iloc[:200]['longitude'].mean()], zoom_start=10)
# Add a circle marker for each row in the DataFrame
for index, row in data.iloc[:200].iterrows():
    # Define the location of the circle marker
    location = [row['latitude'], row['longitude']]
    # Define the pop-up label for the circle marker
    popup_label = "Population: " + str(row['population'])
    # Add the circle marker to the map with customizations
    folium.CircleMarker(location=location, radius=5, color='green',
    fill_color='red', popup=popup_label).add_to(m)
    # Display the map
m.save("1.html")
m
```

Make this Notebook Trusted to load map: File -> Trust Notebook

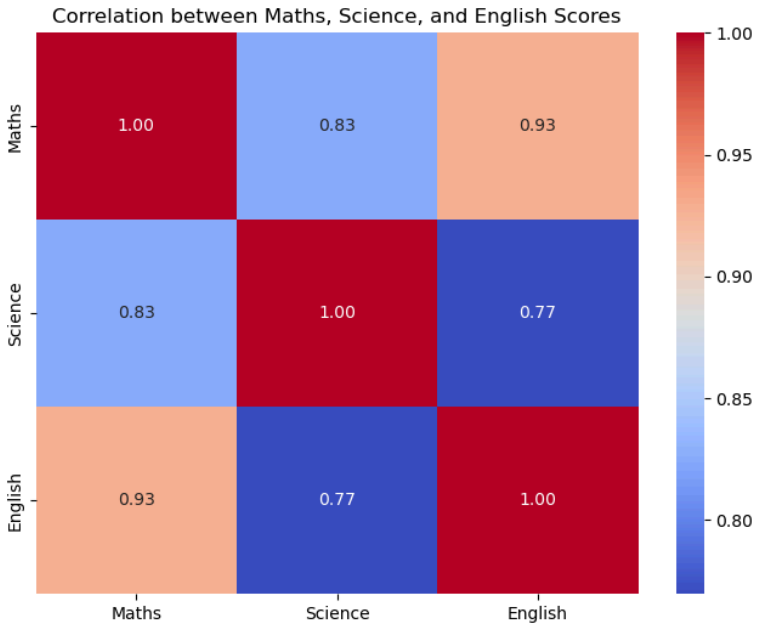


79 The file "student_scores.csv" contains the marks scored by a group of students in three subjects: Maths, Science, and English. Each row contains the name of the student, their score in Maths, Science, and English. Create a pandas DataFrame from this data and create a heatmap to visualize the correlations between the scores in these three subjects. Take Dataset from below:

https://raw.githubusercontent.com/kavit88/Data-Sets/main/student_scores.csv

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset into a DataFrame
url = 'https://raw.githubusercontent.com/kavit88/Data-Sets/main/student_scores.csv'
df = pd.read_csv(url)
# Compute the correlation matrix
correlation_matrix = df[['Maths', 'Science', 'English']].corr()
# Visualize the correlations using a heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation between Maths, Science, and English Scores')
plt.show()
```

```
First few rows of the DataFrame:
    Name  Maths  Science  English
0   John     85       90       78
1  Emily     92       87       91
2   Jack     76       80       82
3  Alice     89       94       86
4    Tom     78       82       76
```

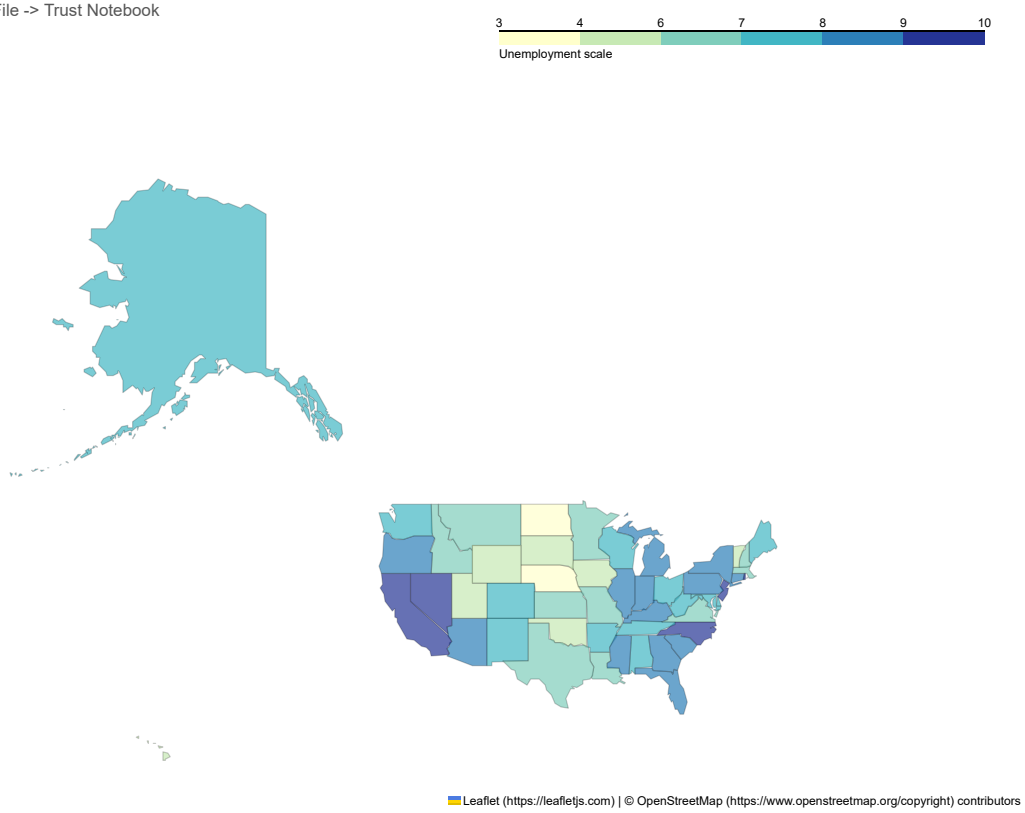


80 You are given a dataset that contains the unemployment rate of different US states for the year 2021. You have to create a choropleth map of the US using the unemployment rate data.

csv file: https://raw.githubusercontent.com/Jovita7/Data-Analysis-and-Visualization/main/US_Unemployment_Oct2012.csv json file: https://raw.githubusercontent.com/Jovita7/Data-Analysis-and-Visualization/main/us-states.json

```
import folium
import pandas as pd
usa_state = folium.Map(location=[48, -102], zoom_start=3)
folium.Choropleth(
    geo_data = 'us-states.json',                  #json
    name ='choropleth',
    data = pd.read_csv("US_Unemployment_Oct2012.csv"),
    columns = ['State', 'Unemployment'], #columns to work on
    key_on ='feature.id',
    fill_color ='YlGnBu',     #I passed colors Yellow,Green,Blue
    fill_opacity = 0.7,
    line_opacity = 0.2,
   legend_name = "Unemployment scale"
).add_to(usa_state)
usa_state
```

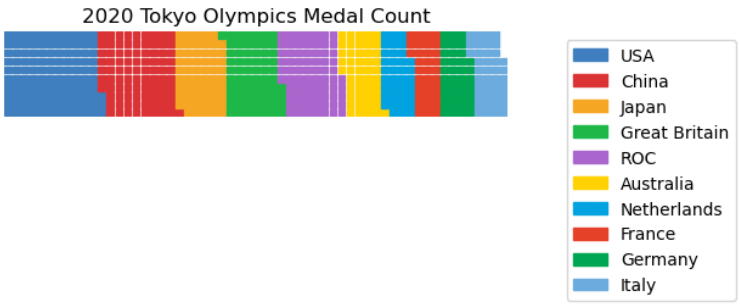
Make this Notebook Trusted to load map: File -> Trust Notebook



83 "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}"

```
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', 'bbox_to_anchor': (1.1, 1)}
)
# Add title
plt.title('2020 Tokyo Olympics Medal Count')
# Show the chart
plt.show()
```

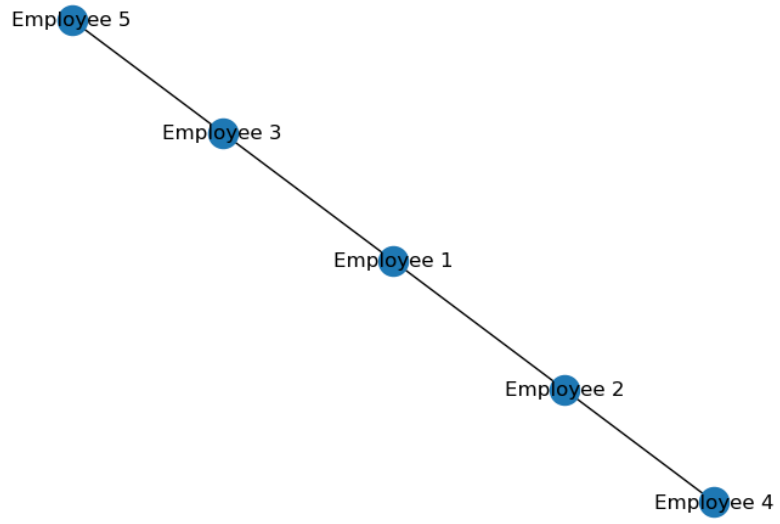


84 "You have been hired as a network analyst by a company to analyze the social network of their employees. The company has provided you with the following data:

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."

```
import networkx as nx
import matplotlib.pyplot as plt
# Create an empty undirected graph
G = nx.Graph()
# 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(G, labels=labels, with_labels=True)
plt.show()
```

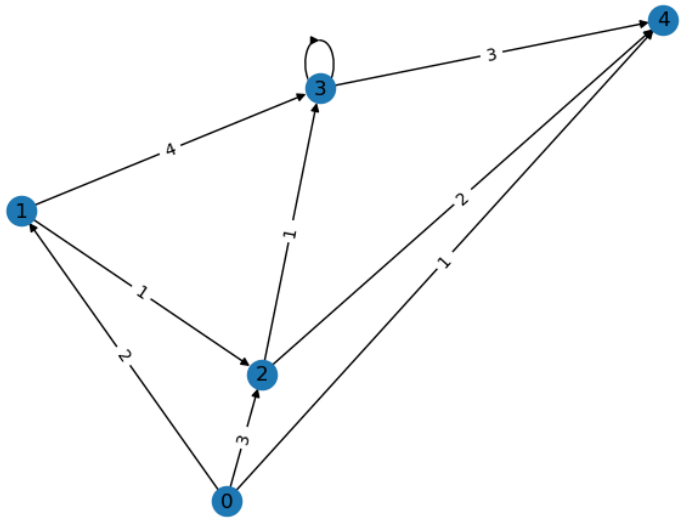


86 "You have been hired as a network analyst by a company to analyze the social network of their employees. The company has provided you with the following data:

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."

```
Matrix = [
[0, 2, 3, 0, 1],
[0, 0, 1, 4, 0],
[0, 0, 0, 1, 2],
[0, 0, 0, 2, 3],
[0, 0, 0, 0, 0]
]
# Create a directed graph
G = nx.DiGraph()
# Add nodes to the graph
for i in range(len(Matrix)):
    G.add_node(i)
    # Add edges to the graph with labels
    for i in range(len(Matrix)):
        for j in range(len(Matrix)):
            if Matrix[i][j] > 0:
                G.add_edge(i, j, weight=Matrix[i][j])
# Draw the graph with edge labels
pos = nx.spring_layout(G)
nx.draw(G, pos, with_labels=True)
labels = nx.get_edge_attributes(G, 'weight')
nx.draw_networkx_edge_labels(G, pos, edge_labels=labels)
plt.show()
```

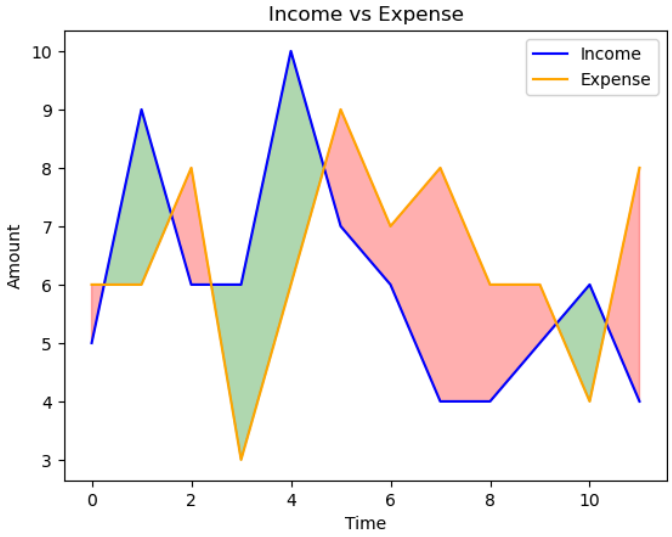


87 Consider the following numpy arrays:

Time=np.arange(12) income=np.array([5,9,6,6,10,7,6,4,4,5,6,4]) expense=np.array([6,6,8,3,6,9,7,8,6,6,4,8])

Use Time array for X-axis and create two separate lines in the same graph with income & expense on Y-axis. Give Appropriate labels. Create an area fill graph between the two lines in such a way that where income is more than expense, are filled with Green and areas where expense is more than income are filled with red.

```
import numpy as np
import matplotlib.pyplot as plt

# Define the numpy arrays
Time = np.arange(12)
income = np.array([5, 9, 6, 6, 10, 7, 6, 4, 4, 5, 6, 4])
expense = np.array([6, 6, 8, 3, 6, 9, 7, 8, 6, 6, 4, 8])

# Plot the lines for income and expense
plt.plot(Time, income, label='Income', color='blue')
plt.plot(Time, expense, label='Expense', color='orange')

# Fill the area between the lines with different colors
plt.fill_between(Time, income, expense, where=(income >= expense), interpolate=True, color='green', alpha=0.3)
plt.fill_between(Time, income, expense, where=(income < expense), interpolate=True, color='red', alpha=0.3)

# Add labels and legend
plt.xlabel('Time')
plt.ylabel('Amount')
plt.title('Income vs Expense')
plt.legend()

# Show the plot
plt.show()
```

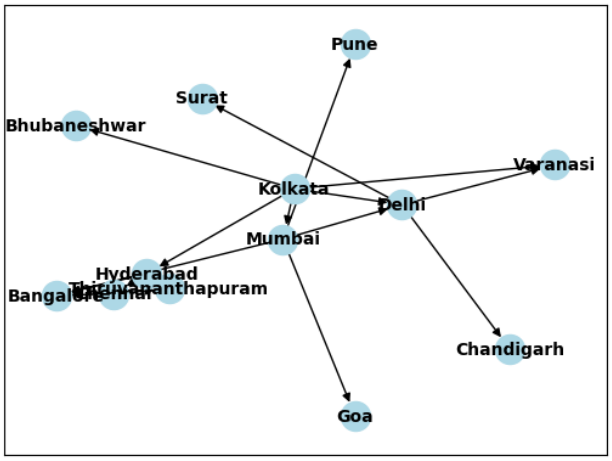


86. "You have been hired by an Airlines company to analyze their routes. The company has provided you following data.

Your task is to create a NetworkX directed graph representing the routes and display it. Figure size should be (15,15), node color should be green, take appropriate node size, edge color should be red.

Data: Kolkata to Mumbai Mumbai to Pune Mumbai to Goa Kolkata to Delhi Kolkata to Bhubaneshwar Mumbai to Delhi Delhi to Chandigarh Delhi to Surat Kolkata to Hyderabad Hyderabad to Chennai Chennai to Thiruvananthapuram Thiruvananthapuram to Hyderabad Kolkata to Varanasi Delhi to Varanasi Mumbai to Bangalore Chennai to Bangalore Hyderabad to Bangalore Kolkata to Guwahati "

```
import matplotlib.pyplot as plt
import networkx as nx

# Create a directed graph object
G = nx.DiGraph()

# Add nodes to the graph with labels
cities = ["Kolkata", "Mumbai", "Pune", "Goa", "Delhi", "Chandigarh", "Surat",
          "Hyderabad", "Chennai", "Thiruvananthapuram", "Varanasi", "Bangalore"]
G.add_nodes_from([(city, {"label": city}) for city in cities])

# Add edges between the nodes, specifying direction (tail -> head)
edges = [
    ("Kolkata", "Mumbai"), ("Mumbai", "Pune"), ("Mumbai", "Goa"),
    ("Kolkata", "Delhi"), ("Kolkata", "Bhubaneshwar"), ("Mumbai", "Delhi"),
    ("Delhi", "Chandigarh"), ("Delhi", "Surat"), ("Kolkata", "Hyderabad"),
    ("Hyderabad", "Chennai"), ("Chennai", "Thiruvananthapuram"),
    ("Thiruvananthapuram", "Hyderabad"), ("Kolkata", "Varanasi"),
    ("Delhi", "Varanasi"), ("Mumbai", "Bangalore"), ("Chennai", "Bangalore"),
    ("Hyderabad", "Bangalore")
]
G.add_edges_from(edges)


# Draw the graph with labels and customize appearance
nx.draw_networkx(G, with_labels=True, font_size=10, node_color='lightblue', edge_color='black', font_weight='bold')

# Display the graph
plt.show()
```



87 Using 'supermarket_sales.csv' file do the following operations and give required answer by using proper programming process.

1). Load the dataset into a pandas DataFrame and read first 8 rows. 2). Check for missing values and fill it by mean values of that particular column if any. 3). Find the number of orders which have 'Quantity' less than 3 and which have (either 'Rating' greater than 8.5 or 'Total' greater than 600). 4). Find the sum of 'Total' purchasing price spent by Member and Normal 'Customer type'. 5). Find the percentage of total of 'gross income' based on the different 'Payment' methods used by customers. (Ewallet, Cash and Credit card) 6). Analyze the purchasing behavior of male and female customers using 'Gender' column. Find their average purchase prices using 'Total' column. 7). Create a scatter plot that shows the relationship between total amount spent and rating. (keep '+' marker, with marker size 100 and green color). 8). Create a box plot that shows the distribution of 'Rating' and 'Quantity'. And comment about outliers in both columns. 9). Visualize with parallel co-ordinates for 'Unit price', 'Total', 'cogs' columns' data with respect to 'Product line'.

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
sales_data=pd.read_csv("supermarket_sales.csv")
sales_data.head(8)
```

	Invoice ID	Branch	City	Customer type	Gender	Product line	Unit price	Quantity	Tax 5%	Total	Date	Time	Payment	cogs	gross margin percentage	gross income
0	750-67-8428	A	Yangon	Member	Female	Health and beauty	74.69	7	26.1415	548.9715	01-05-2019	13:08	Ewallet	522.83	4.761905	26.1415
1	226-31-3081	C	Naypyitaw	Normal	Female	Electronic accessories	15.28	5	3.8200	80.2200	03-08-2019	10:29	Cash	76.40	4.761905	3.8200
2	631-41-3108	A	Yangon	Normal	Male	Home and lifestyle	46.33	7	16.2155	340.5255	03-03-2019	13:23	Credit card	324.31	4.761905	16.2155
3	123-19-1176	A	Yangon	Member	Male	Health and beauty	58.22	8	23.2880	489.0480	1/27/2019	20:33	Ewallet	465.76	4.761905	23.2880
4	373-73-7910	A	Yangon	Normal	Male	Sports and travel	86.31	7	30.2085	634.3785	02-08-2019	10:37	Ewallet	604.17	4.761905	30.2085
5	699-14-3026	C	Naypyitaw	Normal	Male	Electronic accessories	85.39	7	29.8865	627.6165	3/25/2019	18:30	Ewallet	597.73	4.761905	29.8865
6	355-53-5943	A	Yangon	Member	Female	Electronic accessories	68.84	6	20.6520	433.6920	2/25/2019	14:36	Ewallet	413.04	4.761905	20.6520
7	315-22-					Home and										

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
print(df.isna().sum())
sales_data["cogs"].fillna(sales_data["cogs"].mean(), inplace=True)
sales_data["Rating"].fillna(sales_data["Rating"].mean(), inplace=True)
print(df.isna().sum())
```

```
Invoice ID                0
Branch                    0
City                      0
Customer type             0
Gender                    0
Product line              0
Unit price                0
Quantity                  0
Tax 5%                    0
Total                     0
Date                      0
Time                      0
```

```
Payment                   0
cogs                      4
gross margin percentage   0
gross income              0
Rating                    5
dtype: int64
Invoice ID                0
Branch                    0
City                      0
Customer type             0
Gender                    0
Product line              0
Unit price                0
Quantity                  0
Tax 5%                    0
Total                     0
Date                      0
Time                      0
Payment                   0
cogs                      4
gross margin percentage   0
gross income              0
Rating                    5
dtype: int64
C:\Users\VISHAL\AppData\Local\Temp\ipykernel_24052\2946877894.py:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(valu


  sales_data["cogs"].fillna(sales_data["cogs"].mean(), inplace=True)
C:\Users\VISHAL\AppData\Local\Temp\ipykernel_24052\2946877894.py:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(valu


  sales_data["Rating"].fillna(sales_data["Rating"].mean(), inplace=True)
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
filtered_orders = sales_data[(sales_data['Quantity'] < 3) & ((sales_data['Rating'] > 8.5) | (sales_data['Total'] > 600))]
num_filtered_orders = len(filtered_orders)
print("Number of orders meeting the criteria:", num_filtered_orders)
```

```
Number of orders meeting the criteria: 45
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
total_spent_by_type = sales_data[sales_data["Customer type"]=="Normal"]['Total'].sum()
total_spent_by_typ = sales_data[sales_data["Customer type"]=="Member"]['Total'].sum()
print("Total spending by Normal type:")
print(total_spent_by_type)
print("Total spending by Member type:")
print(total_spent_by_typ)
```

```
Total spending by Normal type:
158743.305
Total spending by Member type:
164223.44400000002
```

```
total_spent_by_type = sales_data.groupby('Customer type')['Total'].sum()
print("Total spending by Customer type:")
print(total_spent_by_type)
```

```
Total spending by Customer type:
Customer type
Member    164223.444
Normal    158743.305
Name: Total, dtype: float64
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
# Count occurrences of each payment method
payment_counts = sales_data['Payment'].value_counts()
print(payment_counts)
# Calculate the total gross income
total_income = sales_data['gross income'].sum()

# Calculate the percentage of total gross income for each payment method
percentage_income_by_payment = (payment_counts / len(sales_data)) * 100

# Print the result
print("Percentage of total gross income by Payment method:")
print(percentage_income_by_payment)
```

```
Payment
Ewallet        345
Cash           344
Credit card    311
Name: count, dtype: int64
Percentage of total gross income by Payment method:
Payment
Ewallet        34.5
Cash           34.4
Credit card    31.1
Name: count, dtype: float64
```

```
income_by_payment = sales_data.groupby('Payment')['gross income'].sum()
total_income = sales_data['gross income'].sum()
percentage_income_by_payment = (income_by_payment / total_income) * 100
print("Percentage of total gross income by Payment method:")
print(percentage_income_by_payment)
```

```
Percentage of total gross income by Payment method:
Payment
Cash           34.742453
Credit card    31.200448
Ewallet        34.057099
Name: gross income, dtype: float64
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
average_purchase_by_Male = sales_data[sales_data["Gender"]=="Male"] ['Total'].mean()
print("Average purchase price by Gender:")
print(average_purchase_by_Male)
```

```
Average purchase price by Gender:
310.7892264529058
```

```
average_purchase_by_Female = sales_data[sales_data["Gender"]=="Female"] ['Total'].mean()
print("Average purchase price by Gender:")
print(average_purchase_by_Female)
```

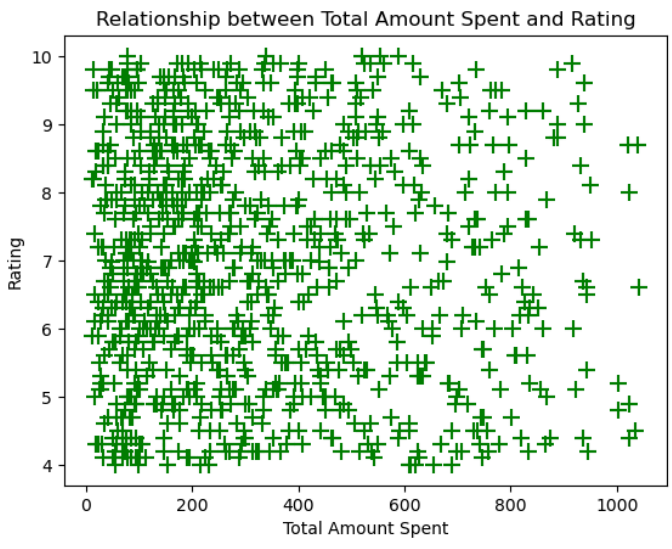
```
Average purchase price by Gender:
335.09565868263473
```

```
average_purchase_by_gender = sales_data.groupby('Gender')['Total'].mean()
print("Average purchase price by Gender:")
print(average_purchase_by_gender)
```

```
Average purchase price by Gender:
Gender
Female    335.095659
Male      310.789226
Name: Total, dtype: float64
```

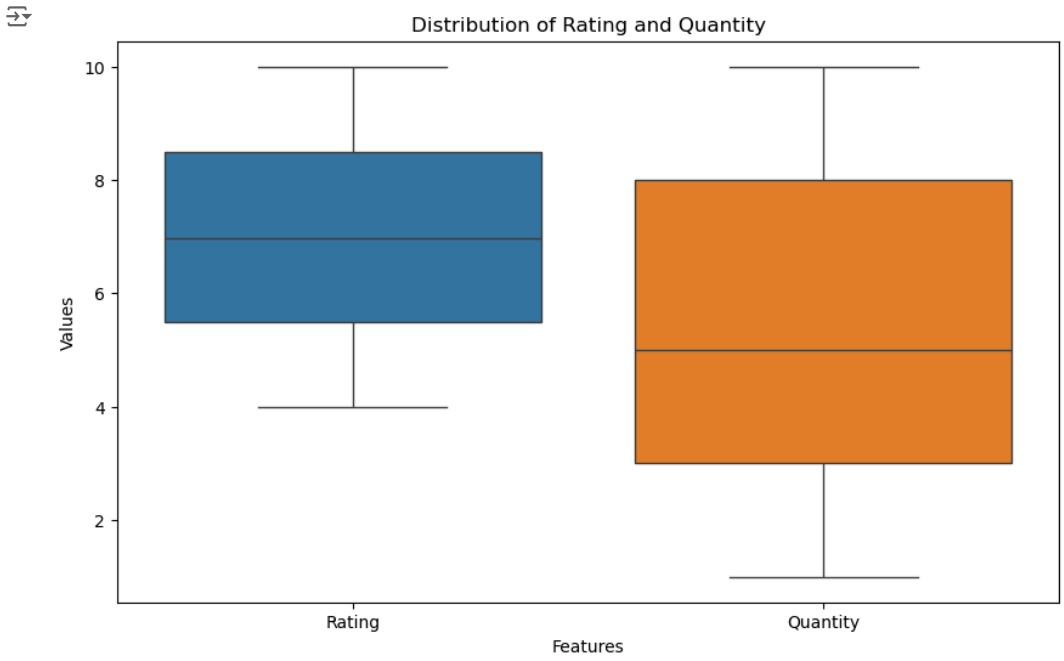
Unsupported Cell Type. Double-Click to inspect/edit the content.

```
plt.scatter(sales_data['Total'], sales_data['Rating'], marker='+', s=100, color='green')
plt.xlabel('Total Amount Spent')
plt.ylabel('Rating')
plt.title('Relationship between Total Amount Spent and Rating')
plt.show()
```



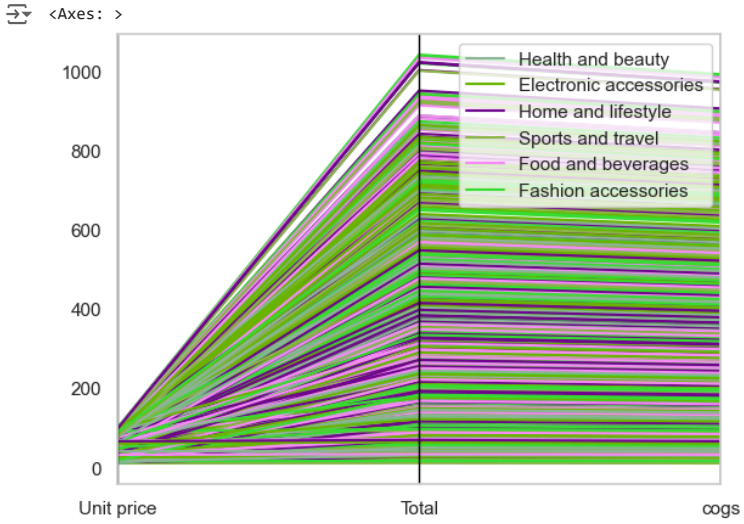
Unsupported Cell Type. Double-Click to inspect/edit the content.

```
import seaborn as sns
plt.figure(figsize=(10, 6))
sns.boxplot(data=sales_data[['Rating', 'Quantity']])
plt.title('Distribution of Rating and Quantity')
plt.xlabel('Features')
plt.ylabel('Values')
plt.show()
```



Unsupported Cell Type. Double-Click to inspect/edit the content.

```
pd.plotting.parallel_coordinates(sales_data,'Product line',["Unit price", "Total", "cogs"])
```



Use the file data.csv which contains 169 rows and 4 columns.

1. Convert this file into pandas Data Frame and Display basic statistics like mean, std, quartiles, etc. for this data frame.
2. Create a correlation table for the data frame and comment about what kind of correlation is there between Duration and Calories?
3. Find whether there any null or NA values, drop all such rows if found in the data frame and print the shape of the data frame after dropping.
4. Prepare a scatter matrix for the following data frame and prepare a parallel coordinates for Duration v/s Pulse, Maxpulse and Calories (all 3 other columns).
5. Do Maxpulse have any outliers? Find using function.
6. Show the outliers using box plot for Maxpulse, width of box plot should be 0.75 and notch should be True.
7. Create a scatter plot for Duration (x-axis) and then Pulse, Maxpulse and Calories (y-axis) with different colors. For each there should be different color and marker.

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df=pd.read_csv("data.csv")
df.describe()
```

	Duration	Pulse	Maxpulse	Calories
count	169.000000	169.000000	169.000000	164.000000
mean	63.846154	107.461538	134.047337	375.790244
std	42.299949	14.510259	16.450434	266.379919
min	15.000000	80.000000	100.000000	50.300000
25%	45.000000	100.000000	124.000000	250.925000
50%	60.000000	105.000000	131.000000	318.600000
75%	60.000000	111.000000	141.000000	387.600000
max	300.000000	159.000000	184.000000	1860.400000

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df.corr(numeric_only=True)
```

	Duration	Pulse	Maxpulse	Calories
Duration	1.000000	-0.155408	0.009403	0.922717
Pulse	-0.155408	1.000000	0.786535	0.025121
Maxpulse	0.009403	0.786535	1.000000	0.203813
Calories	0.922717	0.025121	0.203813	1.000000

```
print("strong relationship with posative value")
```

```
strong relationship with posative value
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
df.isnull().sum()
```

```
Duration    0
Pulse       0
Maxpulse    0
Calories    5
dtype: int64
```

```
df.dropna(inplace=True)
```

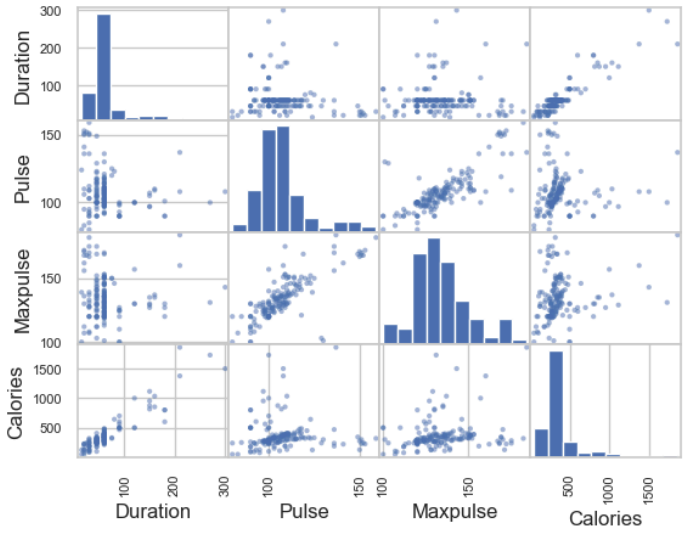
```
df.shape
```

```
(164, 4)
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

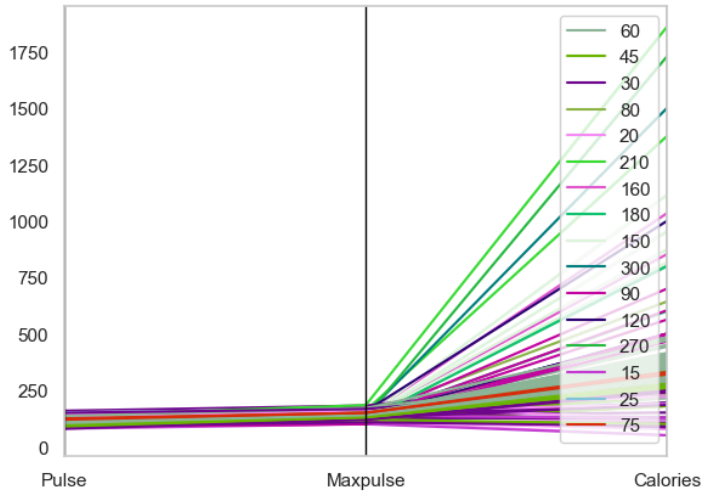
```
pd.plotting.scatter_matrix(df)
```

```
array([[<Axes: xlabel='Duration', ylabel='Duration'>,
        <Axes: xlabel='Pulse', ylabel='Duration'>,
        <Axes: xlabel='Maxpulse', ylabel='Duration'>,
        <Axes: xlabel='Calories', ylabel='Duration'>],
       [<Axes: xlabel='Duration', ylabel='Pulse'>,
        <Axes: xlabel='Pulse', ylabel='Pulse'>,
        <Axes: xlabel='Maxpulse', ylabel='Pulse'>,
        <Axes: xlabel='Calories', ylabel='Pulse'>],
       [<Axes: xlabel='Duration', ylabel='Maxpulse'>,
        <Axes: xlabel='Pulse', ylabel='Maxpulse'>,
        <Axes: xlabel='Maxpulse', ylabel='Maxpulse'>,
        <Axes: xlabel='Calories', ylabel='Maxpulse'>],
       [<Axes: xlabel='Duration', ylabel='Calories'>,
        <Axes: xlabel='Pulse', ylabel='Calories'>,
        <Axes: xlabel='Maxpulse', ylabel='Calories'>,
        <Axes: xlabel='Calories', ylabel='Calories'>]], dtype=object)
```




```
pd.plotting.parallel_coordinates(df,"Duration",["Pulse","Maxpulse","Calories"])
```

```
<Axes: >
```



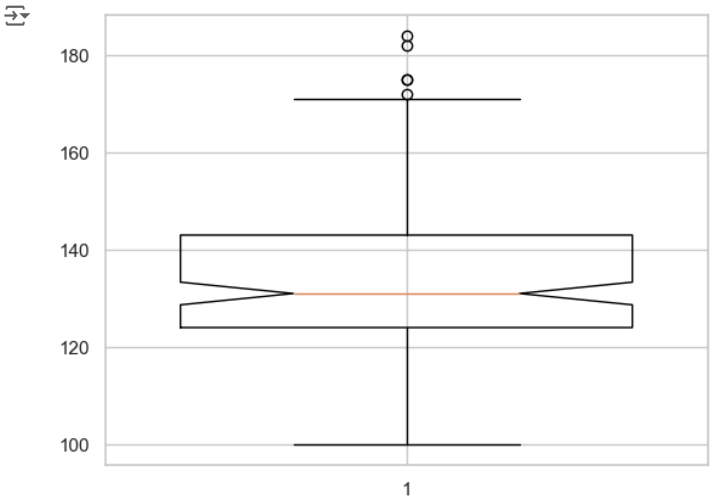
Unsupported Cell Type. Double-Click to inspect/edit the content.

```
def find_outliers_iqr(data_column):
    Q1 = data_column.quantile(0.25)
    Q3 = data_column.quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    outliers = data_column[(data_column < lower_bound) | (data_column > upper_bound)]
    return outliers

maxpulse_outliers = find_outliers_iqr(df['Maxpulse'])
print("Outliers in Maxpulse:")
print(maxpulse_outliers)
```

```
Outliers in Maxpulse:
3      175
54     175
58     172
80     182
109    184
Name: Maxpulse, dtype: int64
```

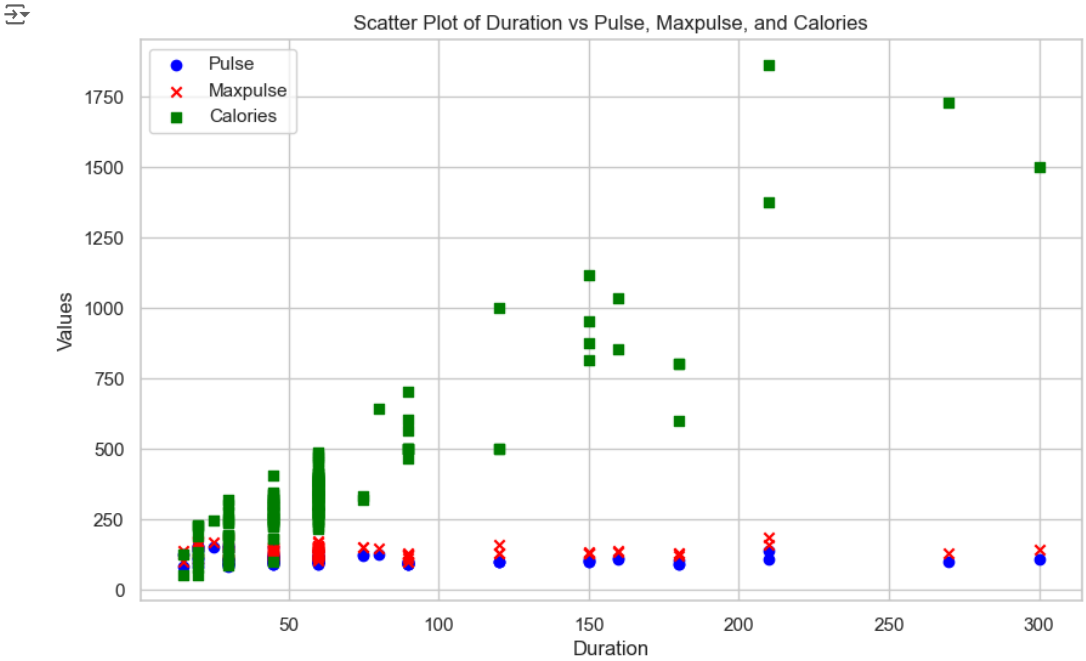
Unsupported Cell Type. Double-Click to inspect/edit the content.

```
plt.boxplot(df.Maxpulse,notch=True,widths=0.75)
plt.show()
```



Unsupported Cell Type. Double-Click to inspect/edit the content.

```
plt.figure(figsize=(10, 6))
plt.scatter(df['Duration'], df['Pulse'], color='blue', marker='o', label='Pulse')
plt.scatter(df['Duration'], df['Maxpulse'], color='red', marker='x', label='Maxpulse')
plt.scatter(df['Duration'], df['Calories'], color='green', marker='s', label='Calories')
plt.xlabel('Duration')
plt.ylabel('Values')
plt.title('Scatter Plot of Duration vs Pulse, Maxpulse, and Calories')
plt.legend()
plt.show()
```



89 The dataset provided in 'kc_house_data.csv' contains house sale prices for King County, which includes Seattle. It includes homes sold between May 2014 and May 2015.

Perform the following tasks : 1) Load the csv to a dataframe named 'house_survey'. 2) Display the first 5 rows of the dataframe. 3) Display the data types of each column. 4) Obtain a statistical summary of the dataframe. 5) Drop the columns "id" and "Unnamed: 0" 6) Check all the null values present in all the columns of the dataframe. 7) Replace the missing values of the column 'bedrooms' with the mean of the column. 8) Replace the missing values of the column 'bathrooms' with the mean of the column. 9) Count the number of houses with unique floor values. 10) Using boxplot determine whether houses with a waterfront view or without a waterfront view have more price outliers. (Mention your answer as comment in the next cell) 11) Use the function regplot in the seaborn library to determine if the feature sqft_above is negatively or

positively correlated with price. (Mention your answer as comment in the next cell). 12) Find the feature other than price that is most correlated with price. (Mention your answer as comment in the next cell).

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
house_survey=pd.read_csv("kc_house_data.csv")
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

```
house_survey.head()
```

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basem
0	7129300520	20141013T000000	221900.0	3	1.00	1180	5650	1.0	0	0	...	7	1180.0	
1	6414100192	20141209T000000	538000.0	3	2.25	2570	7242	2.0	0	0	...	7	2170.0	
2	5631500400	20150225T000000	180000.0	2	1.00	770	10000	1.0	0	0	...	6	770.0	
3	2487200875	20141209T000000	604000.0	4	3.00	1960	5000	1.0	0	0	...	7	1050.0	
4	1954400510	20150218T000000	510000.0	3	2.00	1680	8080	1.0	0	0	...	8	1680.0	

5 rows × 21 columns

```
3) Display the data types of each column.
```

```
  Cell In[123], line 1
    3) Display the data types of each column.
     ^
SyntaxError: unmatched ')'
```

```
house_survey.dtypes
```

```
id                int64
date             object
price           float64
bedrooms          int64
bathrooms       float64
sqft_living       int64
sqft_lot          int64
floors          float64
waterfront        int64
view              int64
condition         int64
grade             int64
sqft_above      float64
sqft_basement     int64
yr_built          int64
yr_renovated      int64
zipcode           int64
lat             float64
long            float64
sqft_living15     int64
sqft_lot15        int64
dtype: object
```

```
4) Obtain a statistical summary of the dataframe.
```

```
  Cell In[125], line 1
    4) Obtain a statistical summary of the dataframe.
     ^
SyntaxError: unmatched ')'
```

```
house_survey.describe(include="all")
```

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	vie
count	2.161300e+04	21613	2.161300e+04	21613.000000	21613.000000	21613.000000	2.161300e+04	21613.000000	21613.000000	21613.00000(
unique	NaN	372	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
top	NaN	20140623T000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
freq	NaN	142	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
mean	4.580302e+09	NaN	5.400881e+05	3.370842	2.114757	2079.899736	1.510697e+04	1.494309	0.007542	0.23430
std	2.876566e+09	NaN	3.671272e+05	0.930062	0.770163	918.440897	4.142051e+04	0.539989	0.086517	0.76631
min	1.000102e+06	NaN	7.500000e+04	0.000000	0.000000	290.000000	5.200000e+02	1.000000	0.000000	0.000000
25%	2.123049e+09	NaN	3.219500e+05	3.000000	1.750000	1427.000000	5.040000e+03	1.000000	0.000000	0.000000
50%	3.904930e+09	NaN	4.500000e+05	3.000000	2.250000	1910.000000	7.618000e+03	1.500000	0.000000	0.000000
75%	7.308900e+09	NaN	6.450000e+05	4.000000	2.500000	2550.000000	1.068800e+04	2.000000	0.000000	0.000000
max	9.900000e+09	NaN	7.700000e+06	33.000000	8.000000	13540.000000	1.651359e+06	3.500000	1.000000	4.000000

11 rows × 21 columns

5) Drop the columns "id" and "Unnamed: 0"

```
  Cell In[127], line 1
    5) Drop the columns "id" and "Unnamed: 0"
     ^
SyntaxError: unmatched ')'
```

```
house_survey.columns
```

```
Index(['id', 'date', 'price', 'bedrooms', 'bathrooms', 'sqft_living',
       'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade',
       'sqft_above', 'sqft_basement', 'yr_built', 'yr_renovated', 'zipcode',
       'lat', 'long', 'sqft_living15', 'sqft_lot15'],
      dtype='object')
```

```
house_survey.drop(house_survey.columns[0], axis=1, inplace = True)
house_survey
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built	yr_re
0	221900.0	3	1.00	1180	5650	1.0	0	0	3	7	1180.0	0	1955	
1	538000.0	3	2.25	2570	7242	2.0	0	0	3	7	2170.0	400	1951	
2	180000.0	2	1.00	770	10000	1.0	0	0	3	6	770.0	0	1933	
3	604000.0	4	3.00	1960	5000	1.0	0	0	5	7	1050.0	910	1965	
4	510000.0	3	2.00	1680	8080	1.0	0	0	3	8	1680.0	0	1987	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
21608	360000.0	3	2.50	1530	1131	3.0	0	0	3	8	1530.0	0	2009	
21609	400000.0	4	2.50	2310	5813	2.0	0	0	3	8	2310.0	0	2014	
21610	402101.0	2	0.75	1020	1350	2.0	0	0	3	7	1020.0	0	2009	
21611	400000.0	3	2.50	1600	2388	2.0	0	0	3	8	1600.0	0	2004	
21612	325000.0	2	0.75	1020	1076	2.0	0	0	3	7	1020.0	0	2008	

21613 rows × 19 columns

6) Check all the null values present in all the columns of the dataframe.

```
house_survey.isnull().sum()
```

```
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
grade            0
sqft_above       2
sqft_basement    0
yr_built         0
yr_renovated     0
zipcode          0
lat              0
long             0
sqft_living15    0
```

```
sqft_lot15       0
dtype: int64
```

7) Replace the missing values of the column 'bedrooms' with the mean of the column.

```
house_survey.bedrooms=house_survey.bedrooms.fillna(house_survey.bedrooms.mean(),inplace=True)
```

8) Replace the missing values of the column 'bathrooms' with the mean of the column.

```
house_survey.bathrooms=house_survey.bathrooms.fillna(house_survey.bathrooms.mean(),inplace=True)
```

9) Count the number of houses with unique floor values.

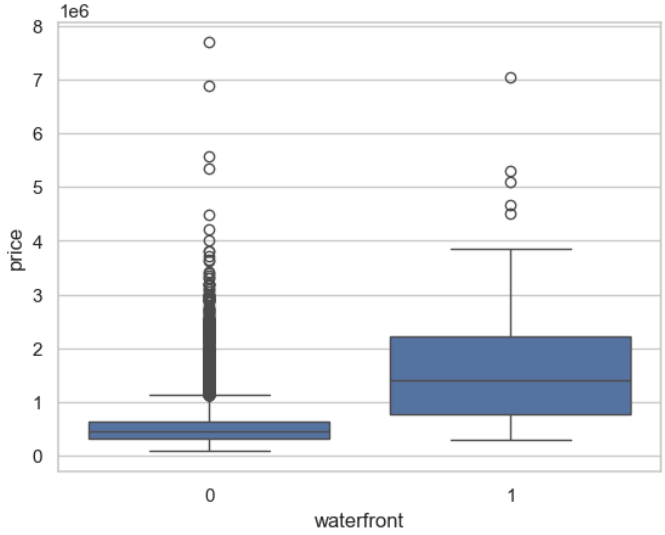
```
df['floors'].value_counts()
```

```
floors
1.0    10680
2.0     8241
1.5     1910
3.0      613
2.5      161
3.5        8
Name: count, dtype: int64
```

10) Using boxplot determine whether houses with a waterfront view or without a waterfront view have more price outliers. (Mention your answer as comm

```
sns.boxplot(x="waterfront", y="price", data=df)
```

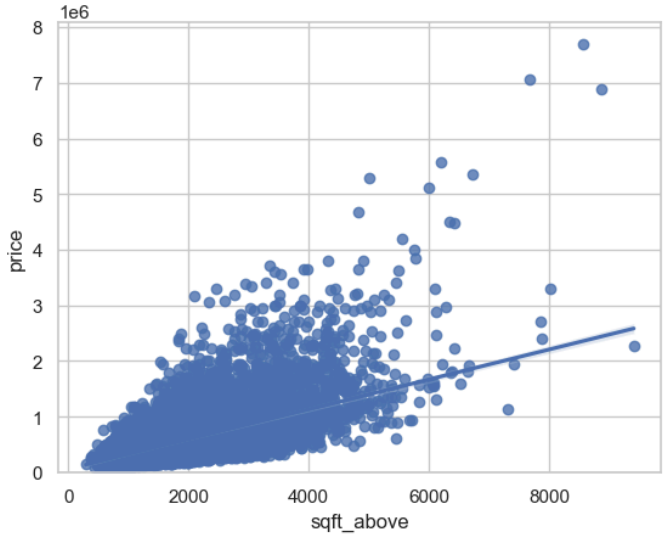
```
<Axes: xlabel='waterfront', ylabel='price'>
```



Unsupported Cell Type. Double-Click to inspect/edit the content.

```
sns.regplot(x="sqft_above", y="price", data=df)
plt.ylim(0,)
```

```
(0.0, 8081250.0)
```



12) Find the feature other than price that is most correlated with price. (Mention your answer as comment in the next cell)

```
house_survey.corr(numeric_only=True)
```

	price	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built	yr_reno
price	1.000000	0.702035	0.089661	0.256794	0.266369	0.397293	0.036362	0.667434	0.605567	0.323816	0.054012	0.1
sqft_living	0.702035	1.000000	0.172826	0.353949	0.103818	0.284611	-0.058753	0.762704	0.876644	0.435043	0.318049	0.0
sqft_lot	0.089661	0.172826	1.000000	-0.005201	0.021604	0.074710	-0.008958	0.113621	0.183511	0.015286	0.053080	0.0
floors	0.256794	0.353949	-0.005201	1.000000	0.023698	0.029444	-0.263768	0.458183	0.523899	-0.245705	0.489319	0.0
waterfront	0.266369	0.103818	0.021604	0.023698	1.000000	0.401857	0.016653	0.082775	0.072074	0.080588	-0.026161	0.0
view	0.397293	0.284611	0.074710	0.029444	0.401857	1.000000	0.045990	0.251321	0.167648	0.276947	-0.053440	0.1
condition	0.036362	-0.058753	-0.008958	-0.263768	0.016653	0.045990	1.000000	-0.144674	-0.158206	0.174105	-0.361417	-0.0
grade	0.667434	0.762704	0.113621	0.458183	0.082775	0.251321	-0.144674	1.000000	0.755924	0.168392	0.446963	0.0
sqft_above	0.605567	0.876644	0.183511	0.523899	0.072074	0.167648	-0.158206	0.755924	1.000000	-0.051976	0.423915	0.0
sqft_basement	0.323816	0.435043	0.015286	-0.245705	0.080588	0.276947	0.174105	0.168392	-0.051976	1.000000	-0.133124	0.0
yr_built	0.054012	0.318049	0.053080	0.489319	-0.026161	-0.053440	-0.361417	0.446963	0.423915	-0.133124	1.000000	-0.2
yr_renovated	0.126434	0.055363	0.007644	0.006338	0.092885	0.103917	-0.060618	0.014414	0.023283	0.071323	-0.224874	1.0
zipcode	-0.053203	-0.199430	-0.129574	-0.059121	0.030285	0.084827	0.003026	-0.184862	-0.261192	0.074845	-0.346869	0.0
lat	0.307003	0.052529	-0.085683	0.049614	-0.014274	0.006157	-0.014941	0.114084	-0.000810	0.110538	-0.148122	0.0
long	0.021626	0.240223	0.229521	0.125419	-0.041910	-0.078400	-0.106500	0.198372	0.343800	-0.144765	0.409356	-0.0
sqft_living15	0.585379	0.756420	0.144608	0.279885	0.086463	0.280439	-0.092824	0.713202	0.731871	0.200355	0.326229	-0.0
sqft_lot15	0.082447	0.183286	0.718557	-0.011269	0.030703	0.072575	-0.003406	0.119248	0.194048	0.017276	0.070958	0.0

```
print("sqft_living: strong relationship")
```

```
sqft_living: strong relationship
```

90 For the given dataset – iris.csv, perform following exploratory data analysis using python -

Use comment feature to answer appropriate questions –

a) Load dataset into jupyter notebook using appropriate libraries. Check the datatypes of the dataset attributes. Does the data contain any missing /null values? b) Extract head and tail of the dataset using appropriate methods. c) Summarize statistical figures (i.e. mean, median, percentiles) in one table using appropriate method. d) Create correlation table of all variables. What can you infer about relation between petal length and sepal length? e) Create parallel coordinate plot of iris dataset. What can you infer about petal length and petal width? f) Create box plot of sepal width. Visualizing the plot, answer whether the sepal width data contains any outliers. g) Create cross tabulation of sepal length and petal width attributes. What does the table represent? h) Create scatter matrix of the dataset. i) Create a new column called 'SepalLengthSize' which contains "High" if sepal length ≥ 5 or "Low" if sepal length < 5.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import scatter_matrix

# a) Load dataset into jupyter notebook using appropriate libraries. Check the datatypes of the dataset attributes. Does the data contain any missing
# Load the dataset
df = pd.read_csv('iris.csv')

# Check the datatypes
print("Data types of the dataset attributes:")
print(df.dtypes)

# Check for missing/null values
print("Does the data contain any missing/null values?")
print(df.isnull().sum())
```

```
Data types of the dataset attributes:
Id                int64
SepalLengthCm    float64
SepalWidthCm     float64
PetalLengthCm    float64
PetalWidthCm     float64
Species           object
dtype: object
Does the data contain any missing/null values?
Id               0
SepalLengthCm    0
SepalWidthCm     0
```

```
PetalLengthCm    0
PetalWidthCm     0
Species          0
dtype: int64
```

```
# b) Extract head and tail of the dataset using appropriate methods.
# Head of the dataset
print("Head of the dataset:")
print(df.head())

# Tail of the dataset
print("Tail of the dataset:")
print(df.tail())
```

In [1]:
```
# 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 [2]:
```
df=pd.read_csv("diabetes_unclean.csv")
```

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

In [3]:
```
# display the specific statistics or measures that are relevant for object-type
df.describe(include=object)
```

Out[3]:

	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 [4]:
```
df.describe()
```

Out[4]:

	ID	No_Pation	AGE	Urea	Cr	HbA1c	Chol	
count	1009.000000	1.009000e+03	1008.000000	1008.000000	1007.000000	1006.000000	1007.000000	10
mean	339.161546	2.717448e+05	53.620040	5.131094	68.973188	8.284155	4.863873	
std	239.738169	3.365681e+06	8.740975	2.931136	59.813297	2.533576	1.297326	
min	1.000000	1.230000e+02	25.000000	0.500000	6.000000	0.900000	0.000000	
25%	127.000000	2.406500e+04	51.000000	3.700000	48.000000	6.500000	4.000000	
50%	296.000000	3.439900e+04	55.000000	4.600000	60.000000	8.000000	4.800000	
75%	548.000000	4.539000e+04	59.000000	5.700000	73.000000	10.200000	5.600000	
max	800.000000	7.543566e+07	79.000000	38.900000	800.000000	16.000000	10.300000	

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

In [5]:
```
print("number of rows",df.shape[0])
```

```
number of rows 1009
```

In [6]:
```
print("number of columns",df.shape[1])
```

```
number of columns 14
```

5. To check the missing values

In [7]:
```
#To check the missing values
df.isnull().sum()
```

Out[7]:
```
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 [8]:
```
#replace the missing values in the column "HbA1c" with it's mean
df['HbA1c']=df['HbA1c'].fillna(df['HbA1c'].mean())
```

In [9]:
```
#To confirm to change
df.isnull().sum()
```

Out[9]:
```
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 [10]:
```
# Dropping the missing values of other columns
df=df.dropna()
df.isna().sum()
```

Out[10]:
```
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 [9]:
```
#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 [10]:
```
df.CLASS.unique()
```

Out[10]:
```
array(['N', 'N ', 'P', 'Y', 'Y '], dtype=object)
```

In [15]:
```
#in a class columns "N " and "Y " replace with "N" and "Y" respectively
df['CLASS'].iloc[df['CLASS']=="N "]="N"
df['CLASS'].iloc[df['CLASS']=="Y "]="Y"
```

```
C:\Users\VISHAL\AppData\Local\Temp\ipykernel_7652\2054179484.py:2: SettingWithCopyW
arning:
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-docs/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'].iloc[df['CLASS']=="N "]="N"
C:\Users\VISHAL\AppData\Local\Temp\ipykernel_7652\2054179484.py:3: SettingWithCopyW
arning:
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-docs/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'].iloc[df['CLASS']=="Y "]="Y"
```

In [16]:
```
df.CLASS.unique()
```

Out[16]:
```
array(['N', 'P', 'Y'], dtype=object)
```

10. display the correlation between variables

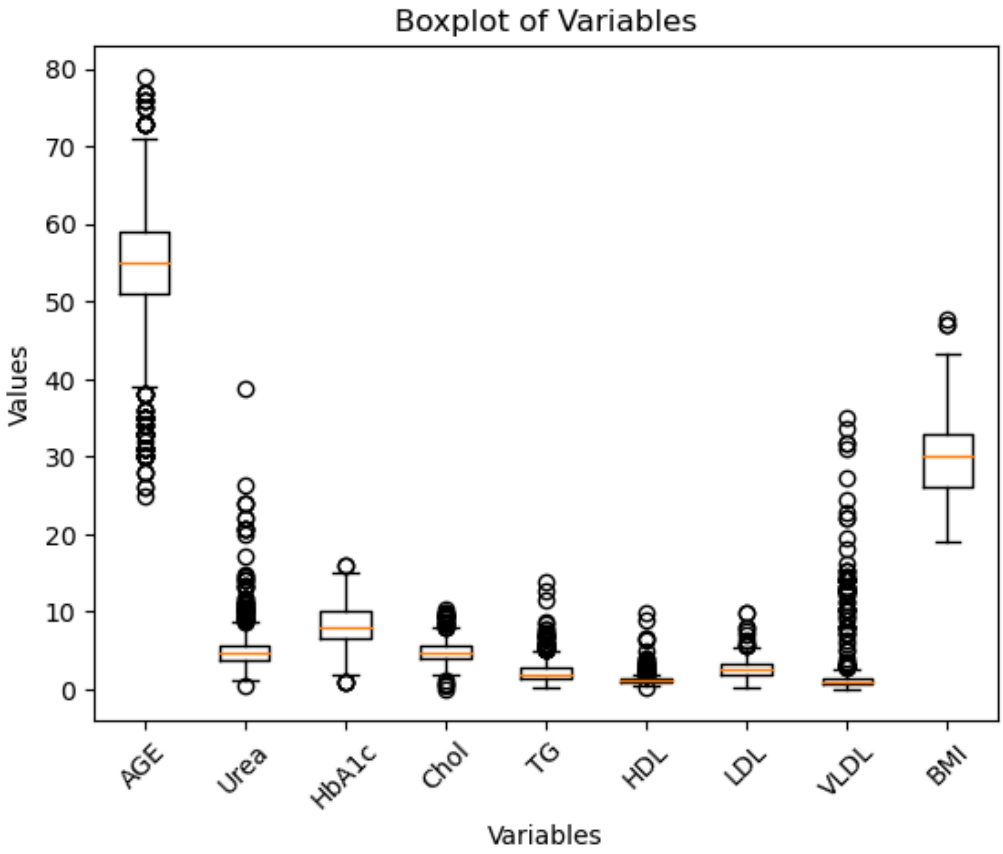
In [30]:
```
#show the correlation between variables?
df.corr()
```

Out[30]:

	ID	No_Pation	AGE	Urea	Cr	HbA1c	Chol	TG	H
ID	1.000000	0.064599	-0.072687	-0.094891	-0.100046	-0.009037	0.045414	-0.054110	0.025
No_Pation	0.064599	1.000000	-0.088870	-0.019061	0.000973	-0.032350	-0.030288	-0.039859	-0.013
AGE	-0.072687	-0.088870	1.000000	0.108613	0.056940	0.384675	0.038966	0.149274	-0.021
Urea	-0.094891	-0.019061	0.108613	1.000000	0.624810	-0.023307	0.001286	0.040939	-0.037
Cr	-0.100046	0.000973	0.056940	0.624810	1.000000	-0.037735	-0.007636	0.056031	-0.023
HbA1c	-0.009037	-0.032350	0.384675	-0.023307	-0.037735	1.000000	0.177676	0.214030	0.030
Chol	0.045414	-0.030288	0.038966	0.001286	-0.007636	0.177676	1.000000	0.318894	0.103
TG	-0.054110	-0.039859	0.149274	0.040939	0.056031	0.214030	0.318894	1.000000	-0.083
HDL	0.025226	-0.013554	-0.021029	-0.037843	-0.023578	0.030455	0.103370	-0.083548	1.000
LDL	-0.065718	-0.003520	0.011496	-0.006673	0.040981	0.011536	0.419237	0.015099	-0.141
VLDL	0.145700	0.113635	-0.090796	-0.011573	0.010328	0.072641	0.076373	0.145825	-0.059
BMI	0.047270	0.017738	0.381176	0.045753	0.054847	0.413130	0.016989	0.110120	0.072

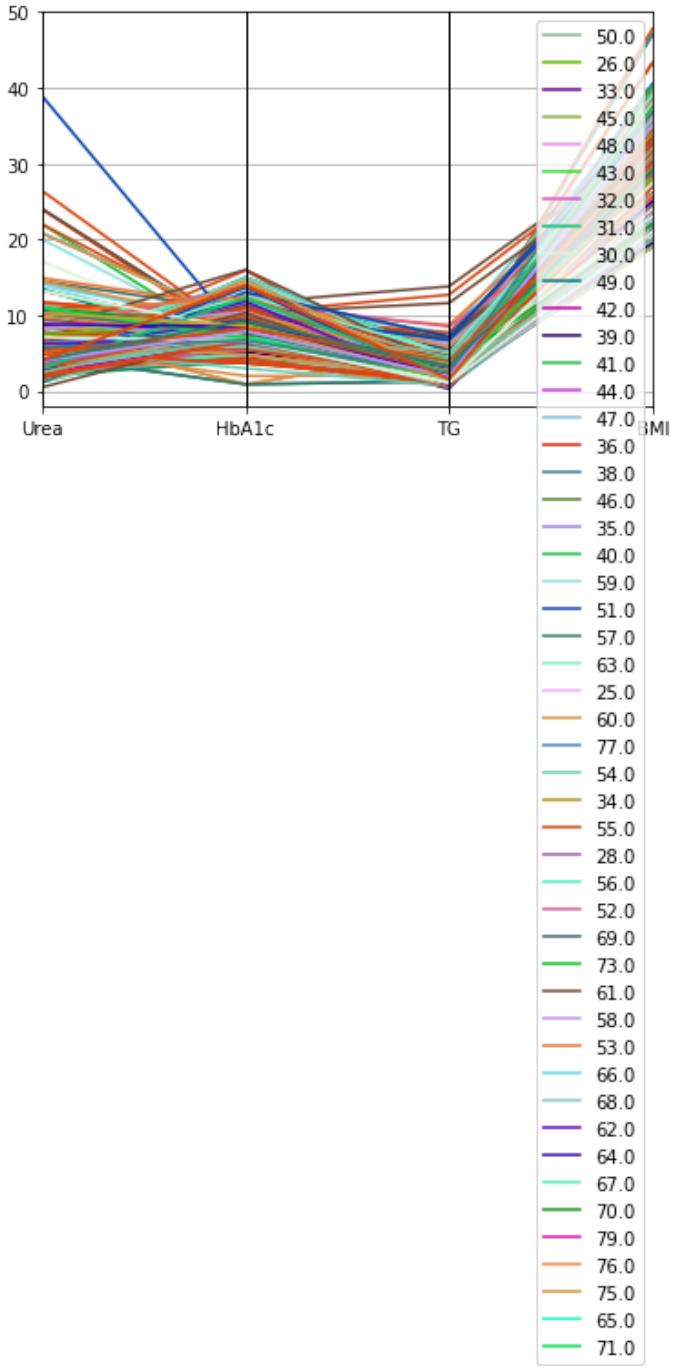
11 checking the outliers in the dataset for the following parameters: 'AGE', 'Urea', 'HbA1c', 'Chol', 'TG', 'HDL', 'LDL', 'VLDL', 'BMI' using box plot with labels and title

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()

```



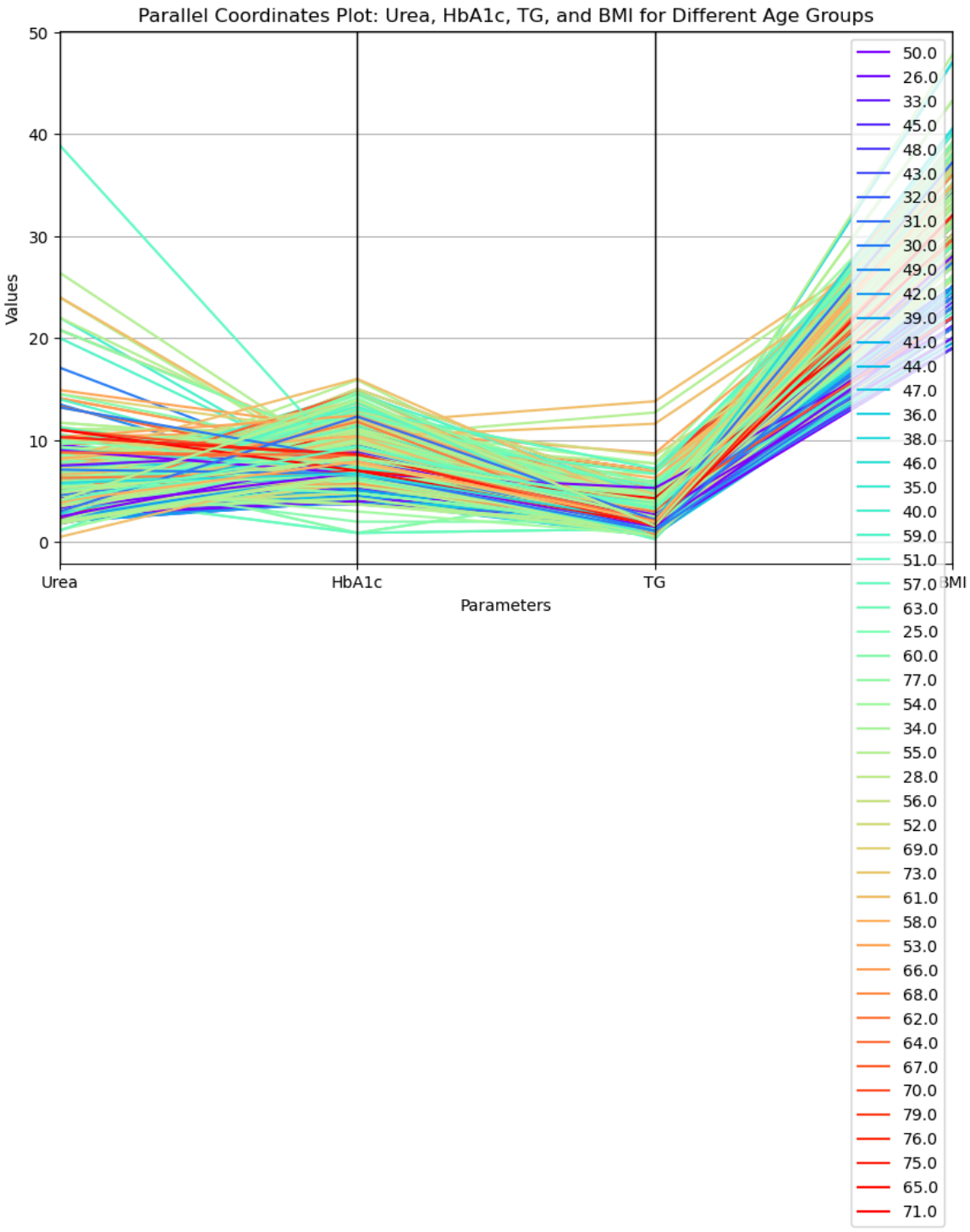
12. Visualized the "Urea", "HbA1c", "TG" and "BMI" parameters for different ages using parallel_coordinates with labels and title

In [12]:
```
# visulize the "Urea","HbA1c","TG" and "BMI" parmaters for different age using p
import matplotlib.pyplot as plt
fig,ax=plt.subplots()
pd.plotting.parallel_coordinates(df,'AGE',["Urea","HbA1c","TG","BMI"])
plt.show()
```



In [29]:

```
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 Ag

# 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 [13]:
```
1 df[df["Gender"]=="f"]
```

Out[13]:

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

In [14]:
```
print(df.shape)
df=df.drop(df[df["Gender"]=="f"].index)
print(df.shape)
```

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

In [56]:
```
pd.crosstab(df["Gender"],df["CLASS"])
```

Out[56]:

Gender	CLASS N	P	Y
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 [15]:
```
# remove the outliers in "HbA1c" columns
import pandas as pd

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

# 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"] <

```

In [16]:
```
df_without_outliers.shape
```

Out[16]: (989, 14)

pQ_3_vha

May 19, 2024

1 129. Write a python program to print Phone number from given string using regular expressions.

[1]:
```
import re

def extract_phone_number(text):
    phone_numbers = re.findall(r'\b\d{10}\b', text)
    return phone_numbers

# Example usage
text = "Contact me at 1234567890 or at 9876543210"
print(extract_phone_number(text))
```

```
['1234567890', '9876543210']
```

2 130 Write a Python program to check that a string contains only a certain set of characters (in this case a-z, A-Z and 0-9) using regular expressions.

[2]:
```
import re

def contains_only_allowed_chars(text):
    return bool(re.findall(r'[a-zA-Z0-9]*', text))

# Example usage
print(contains_only_allowed_chars("Hello123"))
print(contains_only_allowed_chars("Hello 123"))
```

```
True
True
```

3 131 Write a Python program using regular expressions that matches a string that has an a followed by zero or more b's.

[3]:
```
import re

def match_a_followed_by_zero_or_more_bs(text):
    return bool(re.findall(r'ab*', text))

# Example usage
print(match_a_followed_by_zero_or_more_bs("a"))
print(match_a_followed_by_zero_or_more_bs("abbb"))
print(match_a_followed_by_zero_or_more_bs("ac"))
```

```
True
True
True
```

4 132 Write a Python program that matches a string that has an 'a' followed by one or more b's using regular expressions.

[4]:
```
import re

def match_a_followed_by_one_or_more_bs(text):
    return bool(re.findall(r'ab+', text))

# Example usage
print(match_a_followed_by_one_or_more_bs("a"))
print(match_a_followed_by_one_or_more_bs("ab"))
print(match_a_followed_by_one_or_more_bs("abbb"))
```

```
False
True
True
```

5 133 Write a Python program that matches a string that has an 'a' followed by zero or one 'b' using regular expressions.

[5]:
```
import re

def match_a_followed_by_zero_or_one_b(text):
    return bool(re.findall(r'ab?', text))

# Example usage
print(match_a_followed_by_zero_or_one_b("a"))
print(match_a_followed_by_zero_or_one_b("ab"))
print(match_a_followed_by_zero_or_one_b("abb"))
```

```
True
True
True
```

6 134 Write a Python program that matches a string that has an 'a' followed by three 'b' using regular expressions.

[6]:
```
import re

def match_a_followed_by_three_bs(text):
    return bool(re.findall(r'ab{3}', text))

# Example usage
print(match_a_followed_by_three_bs("ab"))
print(match_a_followed_by_three_bs("abbb"))
print(match_a_followed_by_three_bs("abbbb"))
```

```
False
True
True
```

7 135 Write a Python program to find sequences of lowercase letters joined by an underscore using regular expressions.

[7]:
```
import re
def text_match(text):
        patterns = '^[a-z]+_[a-z]+$'
        if re.search(patterns,  text):
                return 'Found a match!'
        else:
                return('Not matched!')

print(text_match("aab_cbbbc"))
print(text_match("aab_Abbbc"))
print(text_match("Aaab_abbbc"))
```

```
Found a match!
Not matched!
Not matched!
```

8 136 Write a Python program to find the sequences of one upper case letter followed by lower case letters using regular expressions.

[8]:
```
import re
def text_match(text):
        patterns = '[A-Z]+[a-z]+$'
        if re.search(patterns, text):
                return 'Found a match!'
        else:
                return('Not matched!')
print(text_match("AaBbGg"))
print(text_match("Python"))
print(text_match("python"))
print(text_match("PYTHON"))
print(text_match("aA"))
print(text_match("Aa"))
```

```
Found a match!
Found a match!
Not matched!
Not matched!
Not matched!
Found a match!
```

9 137 Write a Python program that matches a word at the end of a string, with optional punctuation using regular expressions.

[9]:
```
import re
def text_match(text):
        patterns = '\w+\S*$'
        if re.search(patterns,  text):
                return 'Found a match!'
        else:
                return('Not matched!')

print(text_match("The quick brown fox jumps over the lazy dog."))
print(text_match("The quick brown fox jumps over the lazy dog. "))
print(text_match("The quick brown fox jumps over the lazy dog "))
```

```
Found a match!
Not matched!
```

```
Not matched!
```

10 138 Write a Python program that matches a word containing 'z' using regular expressions.

[10]:
```
import re
def text_match(text):
        patterns = '\Bz\B'
        if re.search(patterns,  text):
                return 'Found a match!'
        else:
                return('Not matched!')

print(text_match("The quick brown fox jumps over the lazy dog."))
print(text_match("Python Exercises."))
```

```
Found a match!
Not matched!
```

11 139 Write a Python program to match a string that contains only upper and lowercase letters, numbers, and underscores using regular expressions.

[11]:
```
import re
def text_match(text):
        patterns = '^[a-zA-Z0-9_]*$'
        if re.search(patterns,  text):
                return 'Found a match!'
        else:
                return('Not matched!')

print(text_match("The quick brown fox jumps over the lazy dog."))
print(text_match("Python_Exercises_1"))
```

```
Not matched!
Found a match!
```

12 140 Write a Python program that starts each string with a specific number using regular expressions

[12]:
```
import re
def match_num(string):
    text = re.search(r"^5",string)
    if text:
        return True
```

```
    else:
        return False
print(match_num('5-2345861'))
print(match_num('6-2345861'))
```

```
True
False
```

13 141 Write a Python program to remove leading zeros from an IP address using regular expressions.

[13]:
```
import re
ip = "016.08.094.196"
if re.findall("^0",ip):
    ip = re.sub('^0', '', ip)

string = re.sub('\.[0]*', '.', ip)
print(string)
```

```
16.8.94.196
```

14 142 Write a Python program to check for a number at the end of a string using regular expressions.

[14]:
```
import re
def end_num(string):
    text = re.compile(r".*[0-9]$")
    if text.match(string):
        return True
    else:
        return False

print(end_num('abcdef'))
print(end_num('abcdef6'))
```

```
False
True
```

15 143 Write a Python program to search for literal strings within a string using regular expressions.

[15]:
```
import re
patterns = [ 'fox', 'dog', 'horse' ]
text = 'The quick brown fox jumps over the lazy dog.'
for pattern in patterns:
```

```
    print('Searching for "%s" in "%s" ->' % (pattern, text),)
    if re.search(pattern,  text):
        print('Matched!')
    else:
        print('Not Matched!')
```

```
Searching for "fox" in "The quick brown fox jumps over the lazy dog." ->
Matched!
Searching for "dog" in "The quick brown fox jumps over the lazy dog." ->
Matched!
Searching for "horse" in "The quick brown fox jumps over the lazy dog." ->
Not Matched!
```

16 144 Write a Python program to extract year, month and date from an URL.

[16]:
```
import re
def extract_date(url):
        return re.findall(r'/(\d{4})/(\d{1,2})/(\d{1,2})/', url)
url1= "https://www.washingtonpost.com/news/football-insider/wp/2016/09/02/
 ↪odell-beckhams-fame-rests-on-one-stupid-little-ball-josh-norman-tells-author/
 ↪"
print(extract_date(url1))
```

```
[('2016', '09', '02')]
```

17 145 Write a Python program to convert a date of yyyy-mm-dd format to dd-mm-yyyy format using regular expressions.

[17]:
```
import re
def change_date_format(dt):
        return re.sub(r'(\d{4})-(\d{1,2})-(\d{1,2})', '\\3-\\2-\\1', dt)
dt1 = "2026-01-02"
print("Original date in YYY-MM-DD Format: ",dt1)
print("New date in DD-MM-YYYY Format: ",change_date_format(dt1))
```

```
Original date in YYY-MM-DD Format:  2026-01-02
New date in DD-MM-YYYY Format:  02-01-2026
```

18 146 Write a Python program to find all words starting with 'a' or 'e' in a given string using regular expressions.

[18]:
```
import re
# Input.
```

```
text = "The following example creates an ArrayList with a capacity of 50␣
 ↪elements. Four elements are then added to the ArrayList and the ArrayList is␣
 ↪trimmed accordingly."
#find all the words starting with 'a' or 'e'
list = re.findall("[ae]\w+", text)
# Print result.
print(list)
```

```
['example', 'eates', 'an', 'ayList', 'apacity', 'elements', 'elements', 'are',
'en', 'added', 'ayList', 'and', 'ayList', 'ed', 'accordingly']
```

19 147 Write a Python program to abbreviate 'Road' as 'Rd.' in a given string using regular expressions.

[19]:
```
import re
street = '21 Ramkrishna Road'
print(re.sub('Road$', 'Rd.', street))
```

```
21 Ramkrishna Rd.
```

20 148 Write a Python program to replace all occurrences of a space, comma, or dot with a colon using regular expressions.

[20]:
```
# import re
text = 'Python Exercises, PHP exercises.'
print(re.sub("[ ,.]", ":", text))
```

```
Python:Exercises::PHP:exercises:
```

21 149 Write a Python program to replace maximum 2 occurrences of space, comma, or dot with a colon using regular expressions.

[21]:
```
# import re
text = 'Python Exercises, PHP exercises.'
print(re.sub("[ ,.]", ":", text, 2))
```

```
Python:Exercises: PHP exercises.
```

22 150 Write a Python program to convert a camel-case string to a snake-case string using regular expressions.

[22]:
```
def change_case(str):
    res = [str[0].lower()]
    for c in str[1:]:
        if c in ('ABCDEFGHIJKLMNOPQRSTUVWXYZ'):
            res.append('_')
            res.append(c.lower())
        else:
            res.append(c)

    return ''.join(res)

# Driver code
str = "GeeksForGeeks"
print(change_case(str))
```

```
geeks_for_geeks
```

23 151 Write a Python program to remove multiple spaces from a string and store the output in list using regular expressions.

[23]:
```
import re
text1 = 'Python      Exercises'
print("Original string:",text1)
print("Without extra spaces:",re.sub(' +',' ',text1))
```

```
Original string: Python      Exercises
Without extra spaces: Python Exercises
```

24 152 Write a Python program to split a string into uppercase letters using regular expressions.

[24]:
```
import re
text = "PythonTutorialAndExercises"
print(re.findall('[A-Z][^A-Z]*', text))
```

```
['Python', 'Tutorial', 'And', 'Exercises']
```

25 153 Write a Python program to remove the parenthesis area in a string.

[25]:
```
import re
items = ["example (.com)", "w3resource", "github (.com)", "stackoverflow (.
 ↪com)"]
for item in items:
    print(re.sub(r" ?\([^)]+\)", "", item))
```

```
example
w3resource
github
stackoverflow
```

26 154 Write a Python program to insert spaces between words starting with capital letters.

[26]:
```
import re
def capital_words_spaces(str1):
  return re.sub(r"(\w)([A-Z])", r"\1 \2", str1)

print(capital_words_spaces("Python"))
print(capital_words_spaces("PythonExercises"))
print(capital_words_spaces("PythonExercisesPracticeSolution"))
```

```
Python
Python Exercises
Python Exercises Practice Solution
```

27 155 Write a Python program that reads a given expression and evaluates it.

[27]:
```
import re
print("Input number of data sets:")
class c(int):
    def __add__(self,n):
        return c(int(self)+int(n))
    def __sub__(self,n):
        return c(int(self)-int(n))
    def __mul__(self,n):
        return c(int(self)*int(n))
    def __truediv__(self,n):
        return c(int(int(self)/int(n)))

for _ in range(int(input())):
  print("Input an expression:")
```

```
  print(eval(re.sub(r'(\d+)',r'c(\1)',input()[:-1])))
```

```
Input number of data sets:
2
Input an expression:
5+4=
9
Input an expression:
5+8=
13
```

28 156 Write a Python program to remove lowercase substrings from a given string.

[28]:
```
import re
str1 = 'KDeoALOklOOHserfLoAJSIskdsf'
print("Original string:")
print(str1)
print("After removing lowercase letters, above string becomes:")
remove_lower = lambda text: re.sub('[a-z]', '', text)
result =  remove_lower(str1)
print(result)
```

```
Original string:
KDeoALOklOOHserfLoAJSIskdsf
After removing lowercase letters, above string becomes:
KDALOOOHLAJSI
```

29 157 “Write a Python program that checks whether a word starts and ends with a vowel in a given string. Return true if a word matches the condition; otherwise, return false.

Sample Data: (““Red Orange White”“) -> True (””Red White Black”“) -> False (””abcd dkise eosksu”“) -> True”

[29]:
```
import re
def test(text):
    return bool(re.findall('[/^[aeiou]$|^([aeiou]).*\1$/', text))

text ="Red Orange White"
print("Original string:", text)
print("Check beginning and end of a word in the said string with a vowel:")
print(test(text))
text ="Red White Black"
print("\nOriginal string:", text)
print("Check beginning and end of a word in the said string with a vowel:")
```

```
print(test(text))
text ="abcd dkise eosksu"
print("\nOriginal string:", text)
print("Check beginning and end of a word in the said string with a vowel:")
print(test(text))
```

```
Original string: Red Orange White
Check beginning and end of a word in the said string with a vowel:
True

Original string: Red White Black
Check beginning and end of a word in the said string with a vowel:
False

Original string: abcd dkise eosksu
Check beginning and end of a word in the said string with a vowel:
True
```

30 "Write a Python program that takes a string with some words. For two consecutive words in the said string, check whether the first word ends with a vowel and the next word begins with a vowel. If the program meets the condition, return true, otherwise false. Only one space is allowed between the words.

Sample Data: (""These exercises can be used for practice."") -> True (""Following exercises should be removed for practice."") -> False (""I use these stories in my classroom."") -> True"

[30]:
```
import re
def test(text):
    return bool(re.findall('[AEIOUaeiou] [AEIOUaeiou]', text))

text ="These exercises can be used for practice."
print("Original string:", text)
print("Two following words begin and end with a vowel in the said string:")
print(test(text))
text ="Following exercises should be removed for practice."
print("\nOriginal string:", text)
print("Two following words begin and end with a vowel in the said string:")
print(test(text))
text ="I use these stories in my  classroom."
print("\nOriginal string:", text)
print("Two following words begin and end with a vowel in the said string:")
print(test(text))
```

```
Original string: These exercises can be used for practice.
Two following words begin and end with a vowel in the said string:
True
```

```
Original string: Following exercises should be removed for practice.
Two following words begin and end with a vowel in the said string:
False

Original string: I use these stories in my  classroom.
Two following words begin and end with a vowel in the said string:
True
```

31 159"Write a Python Program to find all five-character words in a string.

For example: Input : text = 'The quick brown fox jumps over the lazy dog.' Output : ['quick', 'brown', 'jumps'] "

[31]:
```
import re
text = 'The quick brown fox jumps over the lazy dog.'
print(re.findall(r"\b\w{5}\b", text))
```

```
['quick', 'brown', 'jumps']
```

32 160 "Write a python program that executes following tasks (strictly using regex module)

Given text – " hello welcome to the python exam my email is alice@google.com, world this is bob@meta.com appearing for python exam "

a) Remove leading and trailing spaces of the given text.
b) Replace space between words of the given text by '$' symbol
c) Extract username and host name (i.e. alice,bob,google, meta) in a list "

[32]:
```
import re

def process_text(text):
    # a) Remove leading and trailing spaces
    text = re.sub(r'^\s+|\s+$', '', text)

    # b) Replace space between words with '$' symbol
    text_with_dollar = re.sub(r'\s+', '$', text)

    # c) Extract username and host name from emails
    email_matches = re.findall(r'(\w+)@(\w+)\.\w+', text)
    usernames_hosts = [item for sublist in email_matches for item in sublist]

    return text, text_with_dollar, usernames_hosts

# Example usage
```

```
text = "   hello welcome to the python exam my email is alice@google.com, world␣
 ↪this is bob@meta.com appearing for python exam    "
cleaned_text, text_with_dollar, usernames_hosts = process_text(text)

print("Cleaned Text:", cleaned_text)
print("Text with Dollar:", text_with_dollar)
print("Usernames and Hosts:", usernames_hosts)
```

```
Cleaned Text: hello welcome to the python exam my email is alice@google.com,
world this is bob@meta.com appearing for python exam
Text with Dollar: hello$welcome$to$the$python$exam$my$email$is$alice@google.com,
$world$this$is$bob@meta.com$appearing$for$python$exam
Usernames and Hosts: ['alice', 'google', 'bob', 'meta']
```


33 161 "Write a Python Program to find all URLs from a given text. Consider URLs to be of only this format.

http://github.com https://github.com Can Start with either http or https followed by :// domain name dot com

Example:

Text=""Hello all Students must visit at my website https://www.pandasrockstar.com for more information. Also, check out http://www.google.com""

Output: Found URLs: https://www.pandasrockstar.com http://www.google.com "

[33]:
```
import re

def find_urls(text):
    urls = re.findall(r'https?://[^\s]+', text)
    return urls

# Example usage
text = "Hello all Students must visit at my website https://www.pandasrockstar.
 ↪com for more information. Also, check out http://www.google.com"
print(find_urls(text))
```

```
['https://www.pandasrockstar.com', 'http://www.google.com']
```

[ ]:

[ ]:

You are given a text file named "speech.txt" which contains the transcript of a speech. You need to create a Word Cloud for the most frequent words used in the

speech.

In [26]:

```
import matplotlib.pyplot as plt
from wordcloud import WordCloud

# Read the file
with open("speech.txt", "r") as file:
    text = file.read()

# Generate the word cloud
wordcloud = WordCloud(width=800, height=800,
                      background_color='white',
                      min_font_size=10).generate(text)

# Display the word cloud
plt.figure(figsize=(8,8))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()

```

You are given a dataset containing customer reviews of a restaurant. Your task is to create a wordcloud of the most frequent words used in the reviews after removing the stopwords.

In [29]:

```
import pandas as pd
import matplotlib.pyplot as plt
from wordcloud import WordCloud, STOPWORDS

# Load the dataset
df = pd.read_csv("restaurant_reviews.csv")

# Join all the reviews into a single string
text = " ".join(review for review in df.restaurant_name)

# Create a set of stopwords
stopwords = set(STOPWORDS)

# Add some additional stopwords specific
#to the restaurant domain
stopwords.update(["restaurant", "food", "service", "menu", "meal"])

# Generate a wordcloud image
wordcloud = WordCloud(stopwords=stopwords, max_words=50,
                      background_color="white").generate(text)

# Display the generated image
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()

```

Suppose you have data on the number of medals won by a country in the 2020 Tokyo