Technological Institute of the Philippines	Quezon City - Computer Engineering
Course Code:	CPE 019
Code Title:	Emerging Technologies in CpE 2
Summer Semester	AY 2024-2024

Hands-on Activity 3.1: Data Analysis	
Name	Ramos, William Laurence M.
Section	CPE32S1
Date Performed:	14/06/24
Date Submitted:	14/06/24
Instructor:	Engr. Roman Richard

Objectives

Part 1: The Dataset

Part 2: Scatterplot Graphs and Correlatable Variables

Part 3: Calculating Correlation with Python

Part 4: Visualizing

Scenario/Background

Correlation is an important statistical relationship that can indicate whether the variable values are linearly related.

In this lab, you will learn how to use Python to calculate correlation. In Part 1, you will setup the dataset.In Part 2, you will learn how to identify if the variables in a given dataset are correlatable. Finally, in Part 3, you will use Python to calculate the correlation between two sets of variable.

Required Resources

- 1 PC with Internet access
- Raspberry Pi version 2 or higher
- Python libraries: pandas, numpy, matplotlib, seaborn
- Datafiles: brainsize.txt

Part 1: The Dataset

You will use a dataset that contains a sample of 40 right-handed Anglo Introductory Psychology students at a large Southwestern university. Subjects took four subtests (Vocabulary, Similarities, Block Design, and Picture Completion) of the Wechsler (1981) Adult Intelligence Scale-Revised. The researchers used Magnetic Resonance Imaging (MRI) to determine the brain size of the subjects. Information about gender and body size (height and weight) are also included. The researchers withheld the weights of two subjects and the height of one subject for reasons of confidentiality. Two simple modifications were applied to the dataset:

1. Replace the quesion marks used to represent the withheld data points described above by the 'NaN' string. The substitution was done because Pandas does not handle the question marks correctly.
2. Replace all tab characters with commas, converting the dataset into a CSV dataset. The prepared dataset is saved as brainsize.txt. Step 1: Loading the Dataset From a File. Before the dataset can be used, it must be loaded onto memory. In the code below, The first line imports the pandas modules and defines pd as a descriptor that refers to the module. The second line loads the dataset CSV file into a variable called brainFile. The third line uses read_csv(), a pandas method, to convert the CSV dataset stored in brainFile into a dataframe. The dataframe is then stored in the brainFrame variable. Run the cell below to execute the described functions.

```
# Code cell 1
import pandas as pd
brainFile = '/content/brainsize.txt'
brainFrame = pd.read_csv(brainFile, delimiter = '\t')
```

Step 2: Verifying the dataframe. To make sure the dataframe has been correctly loaded and created, use the head() method. Another Pandas method, head() displays the first five entries of a dataframe.

```
# Code cell 2
brainFrame.head()
```

	Gender	FSIQ	VIQ	PIQ	Weight	Height	MRI_Count
0	Female	133	132	124	118.0	64.5	816932
1	Male	140	150	124	NaN	72.5	1001121
2	Male	139	123	150	143.0	73.3	1038437
3	Male	133	129	128	172.0	68.8	965353
4	Female	137	132	134	147.0	65.0	951545

Next steps: Generate code with brainFrame View recommended plots

Part 2: Scatterplot Graphs and Correlatable Variables

Step 1: The pandas describe() method. The pandas module includes the describe() method which performs same common calculations against a given dataset. In addition to provide common results including count, mean, standard deviation, minimum, and maximum, describe() is also a great way to quickly test the validity of the values in the dataframe. Run the cell below to output the results computed by describe() against the brainFrame dataframe.

```
# Code cell 3
brainFrame.describe()
```

	FSIQ	VIQ	PIQ	Weight	Height	MRI_Count
count	40.000000	40.000000	40.00000	38.000000	39.000000	4.000000e+01
mean	113.450000	112.350000	111.02500	151.052632	68.525641	9.087550e+05
std	24.082071	23.616107	22.47105	23.478509	3.994649	7.228205e+04
min	77.000000	71.000000	72.00000	106.000000	62.000000	7.906190e+05
25%	89.750000	90.000000	88.25000	135.250000	66.000000	8.559185e+05
50%	116.500000	113.000000	115.00000	146.500000	68.000000	9.053990e+05
75%	135.500000	129.750000	128.00000	172.000000	70.500000	9.500780e+05
max	144.000000	150.000000	150.00000	192.000000	77.000000	1.079549e+06

Double-click (or enter) to edit

Step 2: Scatterplot graphs Scatterplot graphs are important when working with correlations as they allow for a quick visual verification of the nature of the relationship between the variables. This lab uses the Pearson correlation coefficient, which is sensitive only to a linear relationship between two variables. Other more robust correlation methods exist but are out of the scope of this lab. a. Load the required modules. Before graphs can be plotted, it is necessary to import a few modules, namely numpy and matplotlib. Run the cell below to load these modules.

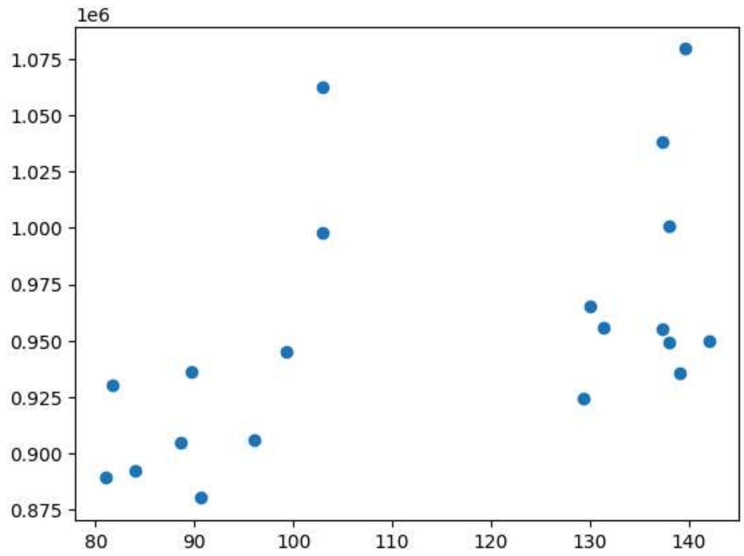
```
# Code cell 4
import numpy as np
import matplotlib.pyplot as plt
```

b. Separate the data. To ensure the results do not get skewed because of the differences in male and female bodies, the dateframe is split into two dataframes: one containing all male entries and another with only female instances. Running the cell below creates the two new dataframes, menDf and womenDf, each one containing the respective entries.

```
# Code cell 5
menDf = brainFrame[(brainFrame.Gender == 'Male')]
womenDf = brainFrame[(brainFrame.Gender == 'Female')]
```

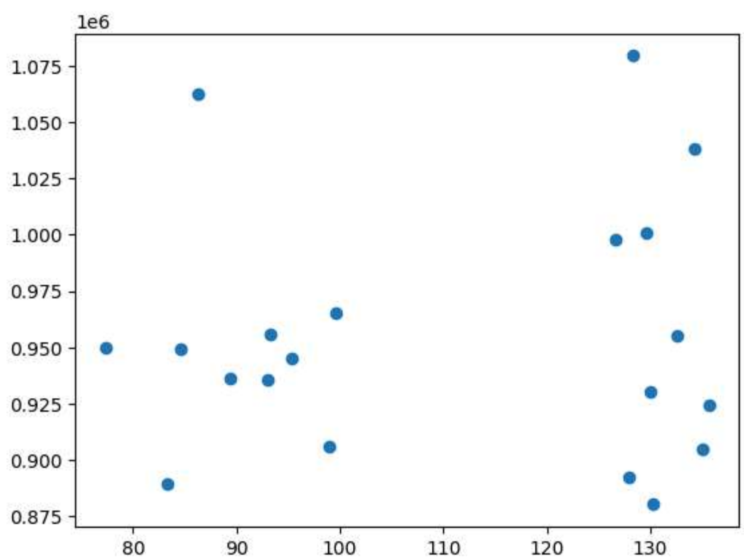
c. Plot the graphs. Because the dataset includes three different measures of intelligence (PIQ, FSIQ, and VIQ), the first line below uses Pandas mean() method to calculate the mean value between the three and store the result in the menMeanSmarts variable. Notice that the first line also refers to the menDf, the filtered dataframe containing only male entries. The second line uses the matplotlib method scatter() to create a scatterplot graph between the menMeanSmarts variable and the MRI_Countattribute. The MRI_Count in this dataset can be thought as of a measure of the physical size of the subjects' brains. The third line simply displays the graph. The fourth line is used to ensure the graph will be displayed in this notebook.

```
# Code cell 6
menMeanSmarts = menDf[["PIQ", "FSIQ", "VIQ"]].mean(axis=1)
plt.scatter(menMeanSmarts, menDf["MRI_Count"])
plt.show()
%matplotlib inline
```



Similarly, the code below creates a scatterplot graph for the women-only filtered dataframe.

```
# Code cell 7
# Graph the women-only filtered dataframe
womenMeanSmarts = womenDf[["PIQ", "FSIQ", "VIQ"]].mean(axis=1)
plt.scatter(womenMeanSmarts, menDf["MRI_Count"])
plt.show()
%matplotlib inline
```



Part 3: Calculating Correlation with Python

Step 1: Calculate correlation against brainFrame. The pandas corr() method provides an easy way to calculate correlation against a dataframe. By simply calling the method against a dataframe, one can get the correlation between all variables at the same time.

```
# Code cell 8
numeric_brainFrame = brainFrame.select_dtypes(include=['number'])
correlation_matrix = numeric_brainFrame.corr(method='pearson')
print(correlation_matrix)
```

```
               FSIQ       VIQ       PIQ    Weight    Height  MRI_Count
FSIQ       1.000000  0.946639  0.934125 -0.051483 -0.086002   0.357641
VIQ        0.946639  1.000000  0.778135 -0.076088 -0.071068   0.337478
PIQ        0.934125  0.778135  1.000000  0.002512 -0.076723   0.386817
Weight    -0.051483 -0.076088  0.002512  1.000000  0.699614   0.513378
Height    -0.086002 -0.071068 -0.076723  0.699614  1.000000   0.601712
MRI_Count  0.357641  0.337478  0.386817  0.513378  0.601712   1.000000
```

```
# Code cell 9
numeric_brainFrame = brainFrame.select_dtypes(include=['number'])

# Define womenDF here before using it.
womenDF = brainFrame[brainFrame['Gender'] == 'Female']

numeric_womenDF = womenDF.select_dtypes(include=['number'])  # Filter non-numeric columns from womenDF
correlation_matrix2 = numeric_womenDF.corr(method='pearson')
print(correlation_matrix2)
```

```
               FSIQ       VIQ       PIQ    Weight    Height  MRI_Count
FSIQ       1.000000  0.955717  0.939382  0.038192 -0.059011   0.325697
VIQ        0.955717  1.000000  0.802652 -0.021889 -0.146453   0.254933
PIQ        0.939382  0.802652  1.000000  0.113901 -0.001242   0.396157
Weight     0.038192 -0.021889  0.113901  1.000000  0.552357   0.446271
Height    -0.059011 -0.146453 -0.001242  0.552357  1.000000   0.174541
MRI_Count  0.325697  0.254933  0.396157  0.446271  0.174541   1.000000
```

And the same can be done for the male-only dataframe:

```
# Code cell 10
numeric_brainFrame = brainFrame.select_dtypes(include=['number'])

# Define menDF here before using it.
menDF = brainFrame[brainFrame['Gender'] == 'Male']

numeric_menDF = menDF.select_dtypes(include=['number'])  # Filter non-numeric columns from womenDF
correlation_matrix2 = numeric_menDF.corr(method='pearson')
print(correlation_matrix2)
```

```
               FSIQ       VIQ       PIQ    Weight    Height  MRI_Count
FSIQ       1.000000  0.944400  0.930694 -0.278140 -0.356110   0.498369
VIQ        0.944400  1.000000  0.766021 -0.350453 -0.355588   0.413105
PIQ        0.930694  0.766021  1.000000 -0.156863 -0.287676   0.568237
Weight    -0.278140 -0.350453 -0.156863  1.000000  0.406542  -0.076875
Height    -0.356110 -0.355588 -0.287676  0.406542  1.000000   0.301543
MRI_Count  0.498369  0.413105  0.568237 -0.076875  0.301543   1.000000
```

Part 4: Visualizing

Step 1: Install Seaborn. To make it easier to visualize the data correlations, heatmap graphs can be used. Based on colored squares, heatmap graphs can help identify correlations in a glance. The Python module named seaborn makes it very easy to plot heatmap graphs. First, run the cell below to download and install the seaborn module.

```
# Code cell 11
!pip install seaborn
```

```
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (0.13.1)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /usr/local/lib/python3.10/dist-packages (from seaborn) (1.25.2)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.10/dist-packages (from seaborn) (2.0.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /usr/local/lib/python3.10/dist-packages (from seaborn) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.2
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (24.1
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.1
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.2->seaborn) (2023.4)
```

```
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.2->seaborn) (2024.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4-
```

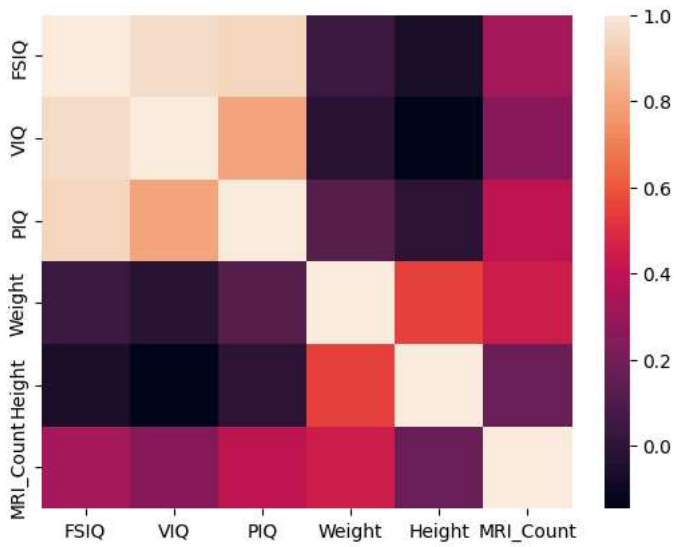
Step 2: Plot the correlation heatmap. Now that the dataframes are ready, the heatmaps can be plotted. Below is a breakdown of the code in the cell below: Line 1: Generates a correlation table based on the womenNoGenderDf dataframe and stores it on wcorr. Line 2: Uses the seaborn heatmap() method to generate and plot the heatmap. Notice that heatmap() takes wcorr as a parameter. Line 3: Use to export and save the generated heatmap as a PNG image. While the line 3 is not active (it has the comment # character preceding it, forcing the interpreter to ignore it), it was kept for informational purposes.

```
# Code cell 12
import seaborn as sns

# Select only the numerical columns for correlation calculation
numerical_womenDf = womenDf.select_dtypes(include=['number'])

wcorr = numerical_womenDf.corr()
sns.heatmap(wcorr)
#plt.savefig('attribute_correlations.png', tight_layout=True)
```

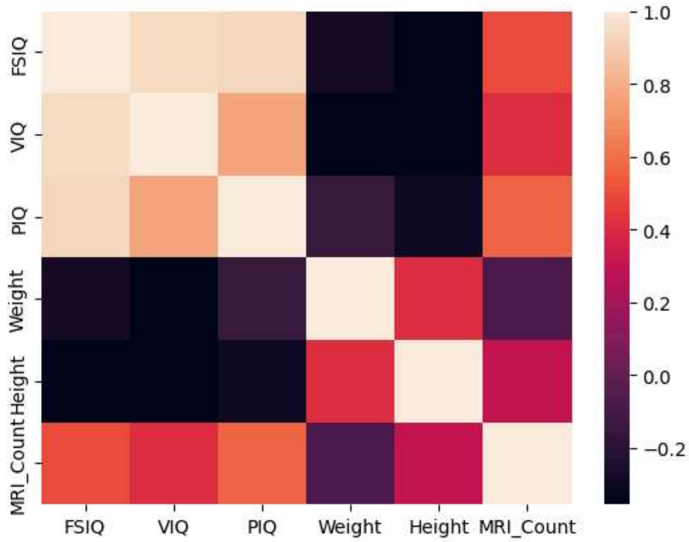
```
<Axes: >
```




```
# Code cell 14
# Select only the numerical columns for correlation calculation
numerical_menDf = menDf.select_dtypes(include=['number'])

mcorr = numerical_menDf.corr()
sns.heatmap(mcorr)
#plt.savefig('attribute_correlations.png', tight_layout=True)
```

```
<Axes: >
```



Many variable pairs present correlation close to zero. What does that mean?

Correlation close to zero between variable pairs indicates no linear relationship, meaning changes in one variable do not predict changes in the other.

Why separate the genders?

Separating genders helps to identify any gender-specific trends or differences that might be masked when data is analyzed collectively.

What variables have stronger correlation with brain size (MRI_Count)? Is that expected? Explain.

Height and weight show stronger correlations with MRI_Count, which is expected as larger body size often correlates with larger brain size due to biological growth patterns.

Supplementary Activity:

Look for (any) real-world dataset and perform exploratory and statistical analysis.

Gender Classification

Context Gender is a social construct. The way males and females are treated differently since birth moulds their behaviour and personal preferences into what society expects for their gender.

This small dataset is designed to provide an idea about whether a person's gender can be predicted with an accuracy significantly above 50% based on their personal preferences.

```
# Code cell 1
import pandas as pd
brainFile = '/content/GenderC.csv'
brainFrame = pd.read_csv(brainFile)
```

```
# Code cell 2
brainFrame.head()
```

	Favorite Color	Favorite Music Genre	Favorite Beverage	Favorite Soft Drink	Gender
0	Cool	Rock	Vodka	7UP/Sprite	F
1	Neutral	Hip hop	Vodka	Coca Cola/Pepsi	F
2	Warm	Rock	Wine	Coca Cola/Pepsi	F
3	Warm	Folk/Traditional	Whiskey	Fanta	F
4	Cool	Rock	Vodka	Coca Cola/Pepsi	F

Next steps: Generate code with brainFrame View recommended plots

```
# Code cell 3
brainFrame.describe()
```

	Favorite Color	Favorite Music Genre	Favorite Beverage	Favorite Soft Drink	Gender
count	66	66	66	66	66
unique	3	7	6	4	2
top	Cool	Rock	Doesn't drink	Coca Cola/Pepsi	F
freq	37	19	14	32	33

```
# Code cell 4
import numpy as np
import matplotlib.pyplot as plt
```

```
 # Code cell 5
menDf = brainFrame[(brainFrame.Gender == 'M')]
womenDf = brainFrame[(brainFrame.Gender == 'F')]
```

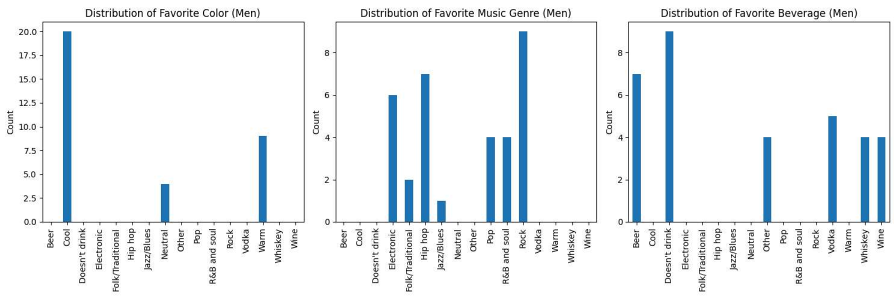
```
 # Code cell 6
import matplotlib.pyplot as plt

# Count the frequency of each category in the selected columns
category_counts = menDf[["Favorite Color", "Favorite Music Genre", "Favorite Beverage"]].apply(pd.value_counts)

# Plot bar charts for each category
fig, axes = plt.subplots(1, 3, figsize=(15, 5))  # 1 row, 3 columns of plots

for i, column in enumerate(["Favorite Color", "Favorite Music Genre", "Favorite Beverage"]):
    category_counts[column].plot(kind='bar', ax=axes[i])
    axes[i].set_title(f'Distribution of {column} (Men)')
    axes[i].set_ylabel('Count')

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



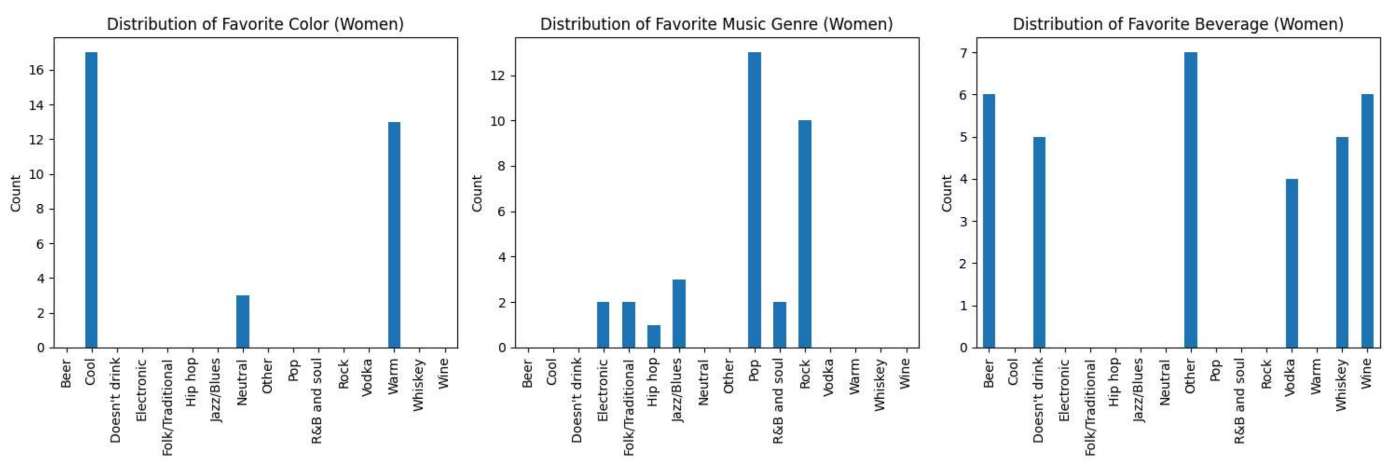

```
 # Code cell 6
import matplotlib.pyplot as plt

# Count the frequency of each category in the selected columns
category_counts = womenDf[["Favorite Color", "Favorite Music Genre", "Favorite Beverage"]].apply(pd.value_counts)

# Plot bar charts for each category
fig, axes = plt.subplots(1, 3, figsize=(15, 5))  # 1 row, 3 columns of plots

for i, column in enumerate(["Favorite Color", "Favorite Music Genre", "Favorite Beverage"]):
    category_counts[column].plot(kind='bar', ax=axes[i])
    axes[i].set_title(f'Distribution of {column} (Women)')
    axes[i].set_ylabel('Count')

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




```
# Code cell 8
numeric_brainFrame = brainFrame.select_dtypes(include=['count'])
correlation_matrix = numeric_brainFrame.corr(method='pearson')
print(correlation_matrix)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-88-67f296f084ba> in <cell line: 2>()
      1 # Code cell 8
----> 2 numeric_brainFrame = brainFrame.select_dtypes(include=['count'])
      3 correlation_matrix = numeric_brainFrame.corr(method='pearson')
      4 print(correlation_matrix)
```

2 frames

```
/usr/local/lib/python3.10/dist-packages/pandas/core/dtypes/common.py in infer_dtype_from_object(dtype)
   1600             pass
   1601
-> 1602     return infer_dtype_from_object(np.dtype(dtype))
   1603
   1604

TypeError: data type 'count' not understood
```

Next steps: Explain error

```
#Code cell 9
```

```
#Code cell 10
```

```
# Code cell 11
!pip install seaborn
```

```
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (0.13.1)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /usr/local/lib/python3.10/dist-packages (from seaborn) (1.25.2)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.10/dist-packages (from seaborn) (2.0.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /usr/local/lib/python3.10/dist-packages (from seaborn) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.2
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (24.1
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.1
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.2->seaborn) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.2->seaborn) (2024.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4-
```

Start coding or generate with AI.

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-87-39f854ada4a3> in <cell line: 2>()
      1 import seaborn as sns
----> 2 dcorr = menDf.corr()
      3 sns.heatmap(dcorr)
      4
```

2 frames