```python
#Install Important Libraries
# !pip install pandas

# Import the Libraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset

import pandas
data1 = pandas.read_csv("C:\\Users\\presc\\OneDrive\\
Fin_wellb05_11_2024.csv")

# Load the dataset

data2 = pd.read_csv("C:\\Users\\presc\\OneDrive\\
State_Data05_11_2024.csv")

#Statistical Report
data1.describe()
```

```
             PUF_ID       sample          fpl        SWB_1
SWB_2  \
count   6394.000000  6394.000000  6394.000000  6394.000000
6394.000000
mean   10892.392712     1.279794     2.658899     5.353769
5.362215
std     1967.854493     0.570187     0.656944     1.500913
1.544942
min     7123.000000     1.000000     1.000000    -4.000000    -
4.000000
25%     9235.250000     1.000000     3.000000     5.000000
5.000000
50%    10901.500000     1.000000     3.000000     6.000000
6.000000
75%    12570.750000     1.000000     3.000000     6.000000
7.000000
max    14400.000000     3.000000     3.000000     7.000000
7.000000

             SWB_3     FWBscore       FWB1_1       FWB1_2       FWB1_3
...  \
count  6394.000000  6394.000000  6394.000000  6394.000000  6394.000000
...
mean      5.432280    56.034094     3.048014     3.191899     2.531279
...
std       1.613876    14.154676     1.235221     1.114130     1.196235
...
```

```
min        -4.000000    -4.000000    -4.000000    -4.000000    -4.000000
...
25%         5.000000    48.000000     2.000000     3.000000     2.000000
...
50%         6.000000    56.000000     3.000000     3.000000     2.000000
...
75%         7.000000    65.000000     4.000000     4.000000     3.000000
...
max         7.000000    95.000000     5.000000     5.000000     5.000000
...

          PPMSACAT       PPREG4       PPREG9        PPT01        PPT25
\
count  6394.000000  6394.000000  6394.000000  6394.000000  6394.000000

mean      0.866124     2.644823     5.145605     0.035815     0.078511

std       0.340545     1.032583     2.529397     0.185843     0.268995

min       0.000000     1.000000     1.000000     0.000000     0.000000

25%       1.000000     2.000000     3.000000     0.000000     0.000000

50%       1.000000     3.000000     5.000000     0.000000     0.000000

75%       1.000000     3.000000     7.000000     0.000000     0.000000

max       1.000000     4.000000     9.000000     1.000000     1.000000


            PPT612      PPT1317      PPT18OV  PCTLT200FPL      finalwt

count  6394.000000  6394.000000  6394.000000  6394.000000  6394.000000

mean      0.129653     0.122928     2.084298    -0.081952     1.000000

std       0.335947     0.328380     0.814345     1.328498     0.585406

min       0.000000     0.000000     1.000000    -5.000000     0.165567

25%       0.000000     0.000000     2.000000     0.000000     0.600582

50%       0.000000     0.000000     2.000000     0.000000     0.845213

75%       0.000000     0.000000     2.000000     0.000000     1.251415

max       1.000000     1.000000     4.000000     1.000000     6.638674


[8 rows x 217 columns]
```

```
data2.describe()
```

	NFCSID	STATEQ	CENSUSDIV	CENSUSREG	A50A
count	2.711800e+04	27118.000000	27118.000000	27118.000000	27118.000000
mean	2.021024e+09	25.719301	5.339627	2.743418	1.540342
std	7.828437e+03	14.863040	2.597738	1.057597	0.498379
min	2.021010e+09	1.000000	1.000000	1.000000	1.000000
25%	2.021017e+09	12.000000	3.000000	2.000000	1.000000
50%	2.021024e+09	26.000000	5.000000	3.000000	2.000000
75%	2.021030e+09	38.000000	8.000000	4.000000	2.000000
max	2.021037e+09	51.000000	9.000000	4.000000	2.000000

	A3Ar_w	A50B	A4A_new_w	A5_2015	A6
count	27118.000000	27118.000000	27118.000000	27118.000000	27118.000000
mean	3.732650	6.974703	1.260196	4.422450	1.890921
std	1.665568	3.435199	0.438750	1.710966	1.165150
min	1.000000	1.000000	1.000000	1.000000	1.000000
25%	2.000000	4.000000	1.000000	3.000000	1.000000
50%	4.000000	7.000000	1.000000	4.000000	2.000000
75%	5.000000	10.000000	2.000000	6.000000	2.000000
max	6.000000	12.000000	2.000000	7.000000	5.000000

	...	M6	M7	M8	M31
count	...	27118.000000	27118.000000	27118.000000	27118.000000
mean	...	15.971716	24.608599	40.859208	30.128033
std	...	34.797002	40.239771	47.100468	43.284233
min	...	1.000000	1.000000	1.000000	1.000000
25%	...	1.000000	3.000000	2.000000	2.000000
50%	...	1.000000	3.000000	3.000000	3.000000
75%	...	2.000000	3.000000	98.000000	98.000000
max	...	99.000000	99.000000	99.000000	99.000000

```
                   M50           M9          M10       wgt_n2
wgt_d2  \
count  27118.000000  27118.000000  27118.000000  27118.000000
27118.000000
mean      33.857364     20.983738     45.376945      1.000000
1.000000
std       45.448122     39.141361     47.901479      0.664533
0.806993
min        1.000000      1.000000      1.000000      0.285976
0.062408
25%        1.000000      1.000000      2.000000      0.462335
0.365107
50%        2.000000      1.000000      2.000000      0.846939
0.783109
75%       98.000000      2.000000     98.000000      1.341735
1.338188
max       99.000000     99.000000     99.000000      5.355691
5.395601

             wgt_s3
count  27118.000000
mean       1.000000
std        0.301259
min        0.270807
25%        0.839477
50%        0.952120
75%        1.093767
max       11.127572

[8 rows x 86 columns]
```

```python
import pandas as pd

# Load data from a CSV file
data1 = pd.read_csv("C:\\Users\\presc\\OneDrive\\
Fin_wellb05_11_2024.csv")

# Rename columns (if needed)
data1 = data1.rename(columns={
    'PUF_ID': 'PUF_ID',
    'sample': 'sample',
    'fpl': 'fpl',
    'SWB_1': 'SWB_1',
    'SWB_2': 'SWB_2',
    'SWB_3': 'SWB_3',
    'FWBscore': 'FWBscore',
    'FWB1_1': 'FWB1_1',
    'FWB1_2': 'FWB1_2',
    'FWB1_3': 'FWB1_3',
    # Continue renaming as needed
```

```python
})

# Calculate Mean, Median, and Mode for selected columns
print("\nMean for 'PUF_ID':")
print(data1['PUF_ID'].mean())
print("\nMedian for 'PUF_ID':")
print(data1['PUF_ID'].median())
print("\nMode for 'PUF_ID':")
print(data1['PUF_ID'].mode()[0])  # Mode returns a series; we access
the first element

print("\nMean for 'sample':")
print(data1['sample'].mean())
print("\nMedian for 'sample':")
print(data1['sample'].median())
print("\nMode for 'sample':")
print(data1['sample'].mode()[0])

print("\nMean for 'fpl':")
print(data1['fpl'].mean())
print("\nMedian for 'fpl':")
print(data1['fpl'].median())
print("\nMode for 'fpl':")
print(data1['fpl'].mode()[0])

# Repeat similar calculations for other columns as needed
```

```
Mean for 'PUF_ID':
10892.392711917422

Median for 'PUF_ID':
10901.5

Mode for 'PUF_ID':
7123

Mean for 'sample':
1.2797935564591805

Median for 'sample':
1.0

Mode for 'sample':
1

Mean for 'fpl':
2.658898967782296

Median for 'fpl':
3.0
```

```
Mode for 'fpl':
3
```

```python
import pandas as pd

# Load data from a CSV file (replace 'your_file.csv' with your file
path)
data2 = pd.read_csv("C:\\Users\\presc\\OneDrive\\
State_Data05_11_2024.csv")

# Clean column names by stripping any leading or trailing spaces
data2.columns = data2.columns.str.strip()

# List of columns in data2 based on your input
columns_to_calculate = [
    'NFCSID', 'STATEQ', 'CENSUSDIV', 'CENSUSREG', 'A50A', 'A3Ar_w',
'A50B', 'A4A_new_w',
    'A5_2015', 'A6', 'M6', 'M7', 'M8', 'M31', 'M50', 'M9', 'M10',
'wgt_n2', 'wgt_d2', 'wgt_s3'
]

# Counter for numbered output
counter = 1

# Calculate and display Mean, Median, and Mode for each column
for column in columns_to_calculate:
    if column in data2.columns:
        mean_value = data2[column].mean()
        median_value = data2[column].median()
        mode_value = data2[column].mode()

        print(f"\n{counter}. Statistics for '{column}':")
        print(f"   Mean: {mean_value}")
        print(f"   Median: {median_value}")
        if not mode_value.empty:
            print(f"   Mode: {mode_value[0]}")
        else:
            print("   Mode: No mode available")
        counter += 1
    else:
        print(f"\n{counter}. Column '{column}' not found in data2.")
        counter += 1
```

```
1. Statistics for 'NFCSID':
   Mean: 2021023559.5
   Median: 2021023559.5
   Mode: 2021010001
```

```
2. Statistics for 'STATEQ':
   Mean: 25.719300833394794
   Median: 26.0
   Mode: 38

3. Statistics for 'CENSUSDIV':
   Mean: 5.339626816136883
   Median: 5.0
   Mode: 5

4. Statistics for 'CENSUSREG':
   Mean: 2.743417656169334
   Median: 3.0
   Mode: 3

5. Statistics for 'A50A':
   Mean: 1.540342208127443
   Median: 2.0
   Mode: 2

6. Statistics for 'A3Ar_w':
   Mean: 3.7326499004351352
   Median: 4.0
   Mode: 6

7. Statistics for 'A50B':
   Mean: 6.974703149199794
   Median: 7.0
   Mode: 12

8. Statistics for 'A4A_new_w':
   Mean: 1.260196179659267
   Median: 1.0
   Mode: 1

9. Statistics for 'A5_2015':
   Mean: 4.422450033188288
   Median: 4.0
   Mode: 4

10. Statistics for 'A6':
   Mean: 1.8909211593775352
   Median: 2.0
   Mode: 1

11. Statistics for 'M6':
   Mean: 15.971716203259827
   Median: 1.0
   Mode: 1
```

```
12. Statistics for 'M7':
   Mean: 24.608599454237037
   Median: 3.0
   Mode: 3

13. Statistics for 'M8':
   Mean: 40.85920790618777
   Median: 3.0
   Mode: 98

14. Statistics for 'M31':
   Mean: 30.128033040784718
   Median: 3.0
   Mode: 2

15. Statistics for 'M50':
   Mean: 33.85736411239767
   Median: 2.0
   Mode: 1

16. Statistics for 'M9':
   Mean: 20.983737738771296
   Median: 1.0
   Mode: 1

17. Statistics for 'M10':
   Mean: 45.37694520244856
   Median: 2.0
   Mode: 98

18. Statistics for 'wgt_n2':
   Mean: 1.0000000000266245
   Median: 0.846938555
   Mode: 1.22670932

19. Statistics for 'wgt_d2':
   Mean: 0.9999999996530348
   Median: 0.783109457
   Mode: 0.250077299

20. Statistics for 'wgt_s3':
   Mean: 1.0000000000012168
   Median: 0.952119531
   Mode: 1.093767206
```

```
import pandas as pd
import numpy as np

# Load data from a CSV file
data1 = pd.read_csv("C:\\Users\\presc\\OneDrive\\
```

```
Fin_wellb05_11_2024.csv")

# Print column names to check for exact names and any discrepancies
print("Column names in data1:")
print(data1.columns)

# Update this list with the exact column names found in the previous
output
columns_to_calculate_data1 = [
    'Frequency', 'Weighted Frequency', 'Percent', 'Lower 95% CL',
'Upper 95% CL'
]

# Convert specified columns to numeric, setting non-numeric values as
NaN
for column in columns_to_calculate_data1:
    if column in data1.columns:
        data1[column] = pd.to_numeric(data1[column], errors='coerce')
    else:
        print(f"Column '{column}' not found in data1 and will be
skipped.")

# Select only the specified numeric columns that exist in data1
data1_numeric = data1[[col for col in columns_to_calculate_data1 if
col in data1.columns]]

# Calculate variance and standard deviation, ignoring NaN values
print("\nVariance for each column in data1:")
print(data1_numeric.var())

print("\nStandard Deviation for each column in data1:")
print(data1_numeric.std())
```

```
Column names in data1:
Index(['PUF_ID', 'sample', 'fpl', 'SWB_1', 'SWB_2', 'SWB_3',
'FWBscore',
       'FWB1_1', 'FWB1_2', 'FWB1_3',
       ...
       'PPMSACAT', 'PPREG4', 'PPREG9', 'PPT01', 'PPT25', 'PPT612',
'PPT1317',
       'PPT18OV', 'PCTLT200FPL', 'finalwt'],
      dtype='object', length=217)
Column 'Frequency' not found in data1 and will be skipped.
Column 'Weighted Frequency' not found in data1 and will be skipped.
Column 'Percent' not found in data1 and will be skipped.
Column 'Lower 95% CL' not found in data1 and will be skipped.
Column 'Upper 95% CL' not found in data1 and will be skipped.

Variance for each column in data1:
```

```
Series([], dtype: float64)

Standard Deviation for each column in data1:
Series([], dtype: float64)
```

```python
import pandas as pd
import numpy as np

# Load data from a CSV file (replace 'your_file.csv' with the actual
file path)
data2 = pd.read_csv("C:\\Users\\presc\\OneDrive\\
State_Data05_11_2024.csv")

# Ensure relevant columns are numeric (replace with correct column
names if needed)
columns_to_calculate = [
    'NFCSID', 'STATEQ', 'CENSUSDIV', 'CENSUSREG', 'A50A', 'A3Ar_w',
'A50B', 'A4A_new_w',
    'A5_2015', 'A6', 'M6', 'M7', 'M8', 'M31', 'M50', 'M9', 'M10',
'wgt_n2', 'wgt_d2', 'wgt_s3'
]

# Convert specified columns to numeric, setting non-numeric values as
NaN
for column in columns_to_calculate:
    data2[column] = pd.to_numeric(data2[column], errors='coerce')

# Select only numeric columns for calculation
data2_numeric = data2[columns_to_calculate]

# Calculate variance and standard deviation, ignoring NaN values
print("Variance for each column in data2:")
print(data2_numeric.var())

print("\nStandard Deviation for each column in data2:")
print(data2_numeric.std())
```

```
Variance for each column in data2:
NFCSID        6.128442e+07
STATEQ        2.209100e+02
CENSUSDIV     6.748240e+00
CENSUSREG     1.118512e+00
A50A          2.483817e-01
A3Ar_w        2.774116e+00
A50B          1.180059e+01
A4A_new_w     1.925012e-01
A5_2015       2.927404e+00
A6            1.357574e+00
M6            1.210831e+03
M7            1.619239e+03
```

```
M8            2.218454e+03
M31           1.873525e+03
M50           2.065532e+03
M9            1.532046e+03
M10           2.294552e+03
wgt_n2        4.416036e-01
wgt_d2        6.512381e-01
wgt_s3        9.075725e-02
dtype: float64

Standard Deviation for each column in data2:
NFCSID       7828.436636
STATEQ         14.863040
CENSUSDIV       2.597738
CENSUSREG       1.057597
A50A            0.498379
A3Ar_w          1.665568
A50B            3.435199
A4A_new_w       0.438750
A5_2015         1.710966
A6              1.165150
M6             34.797002
M7             40.239771
M8             47.100468
M31            43.284233
M50            45.448122
M9             39.141361
M10            47.901479
wgt_n2          0.664533
wgt_d2          0.806993
wgt_s3          0.301259
dtype: float64
```

```
import pandas as pd

# Load data from a CSV file (replace 'your_file.csv' with the actual
file path)
data1 = pd.read_csv("C:\\Users\\presc\\OneDrive\\
Fin_wellb05_11_2024.csv")

# Select only numeric columns in data1
numeric_data1 = data1.select_dtypes(include=['float64', 'int64'])

# Calculate Skewness for each numeric column
print("\nSkewness for numeric columns in data1:")
print(numeric_data1.skew())

# Calculate Kurtosis for each numeric column
print("\nKurtosis for numeric columns in data1:")
print(numeric_data1.kurtosis())
```

```
Skewness for numeric columns in data1:
PUF_ID        -0.031130
sample         1.919704
fpl           -1.694218
SWB_1         -1.203838
SWB_2         -1.297704
                  ...
PPT612         2.205483
PPT1317        2.297279
PPT18OV        0.741445
PCTLT200FPL   -3.012811
finalwt        2.212054
Length: 217, dtype: float64

Kurtosis for numeric columns in data1:
PUF_ID        -1.116662
sample         2.565448
fpl            1.398116
SWB_1          1.802920
SWB_2          2.250429
                  ...
PPT612         2.865052
PPT1317        3.278518
PPT18OV        0.356909
PCTLT200FPL    8.713753
finalwt        8.852834
Length: 217, dtype: float64
```

```python
import pandas as pd

# Load data from a CSV file (replace 'your_file.csv' with the actual
file path)
data2 = pd.read_csv("C:\\Users\\presc\\OneDrive\\
State_Data05_11_2024.csv")

# Select only numeric columns in data2
numeric_data2 = data2.select_dtypes(include=['float64', 'int64'])

# Calculate Skewness for each numeric column
print("\nSkewness for numeric columns in data2:")
print(numeric_data2.skew())

# Calculate Kurtosis for each numeric column
print("\nKurtosis for numeric columns in data2:")
print(numeric_data2.kurtosis())
```

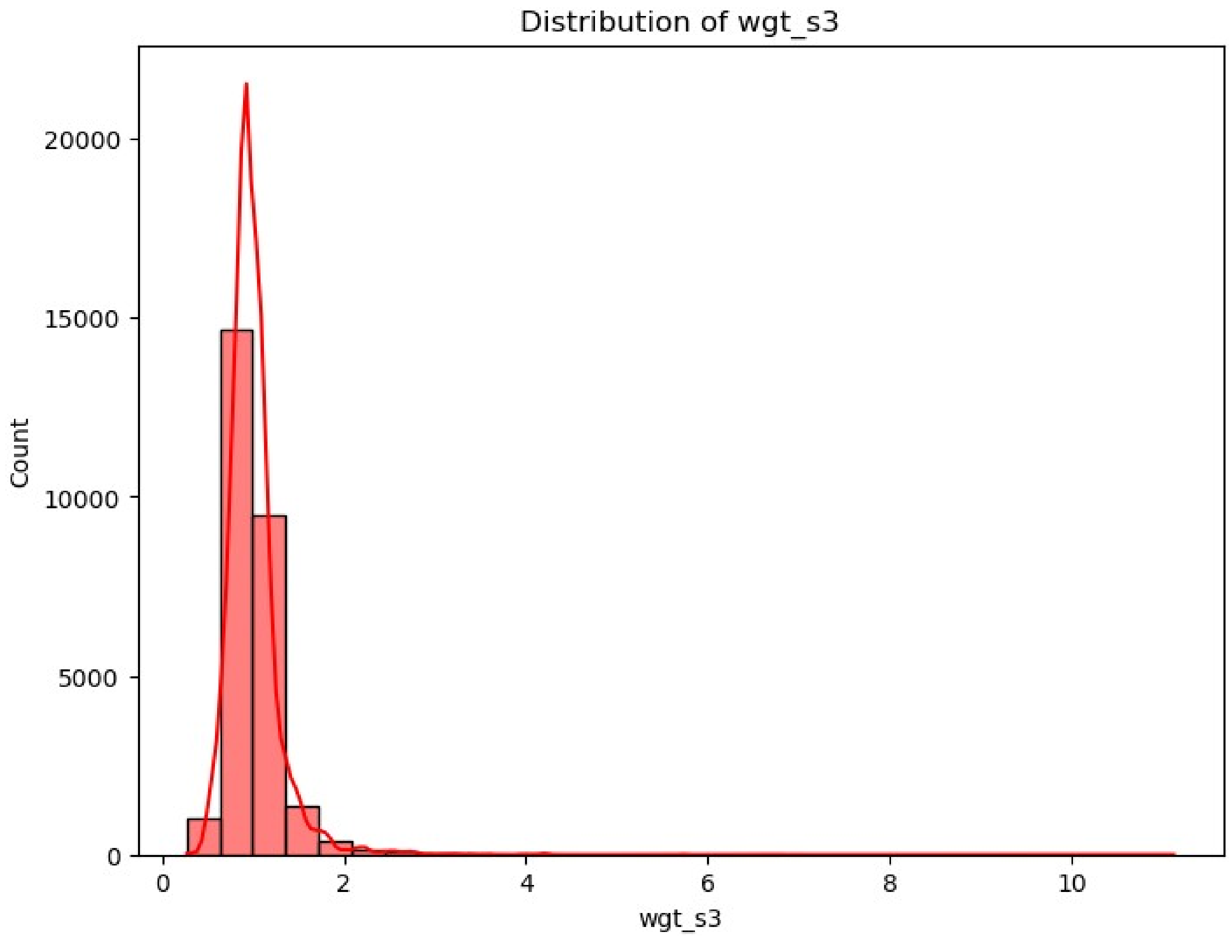
```
Skewness for numeric columns in data2:
NFCSID       0.000000
```

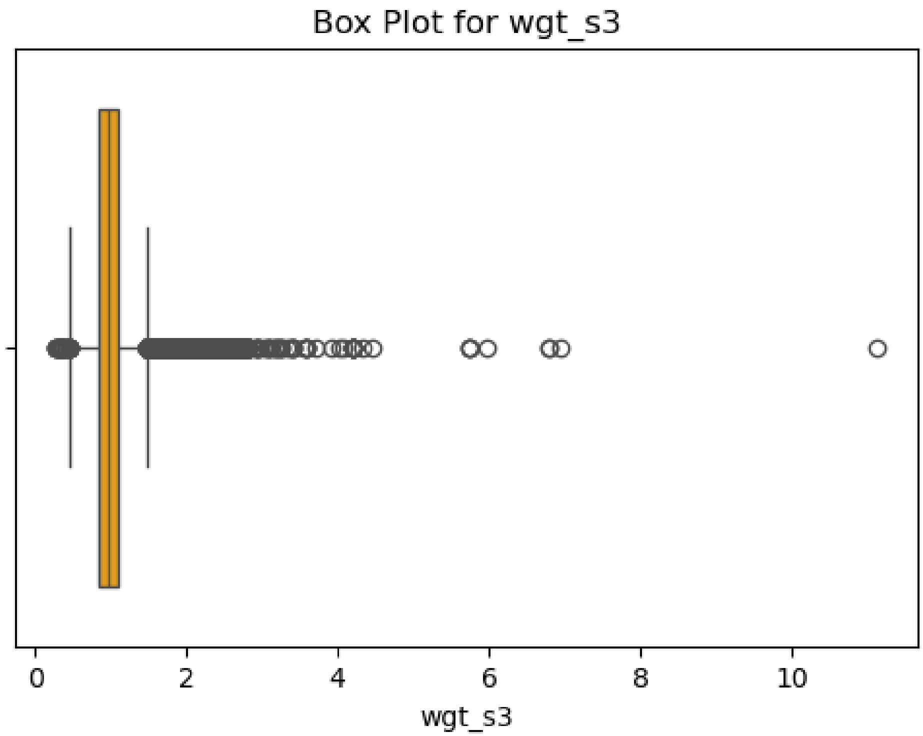
```
STATEQ      -0.008888
CENSUSDIV   -0.122941
CENSUSREG   -0.318593
A50A        -0.161906
               ...
M9           1.460330
M10          0.188802
wgt_n2       1.924560
wgt_d2       1.355259
wgt_s3       5.053243
Length: 86, dtype: float64

Kurtosis for numeric columns in data2:
NFCSID       -1.200000
STATEQ       -1.246895
CENSUSDIV    -1.150638
CENSUSREG    -1.123391
A50A         -1.973932
               ...
M9            0.132876
M10          -1.964325
wgt_n2        6.154866
wgt_d2        1.910834
wgt_s3       84.837405
Length: 86, dtype: float64
```

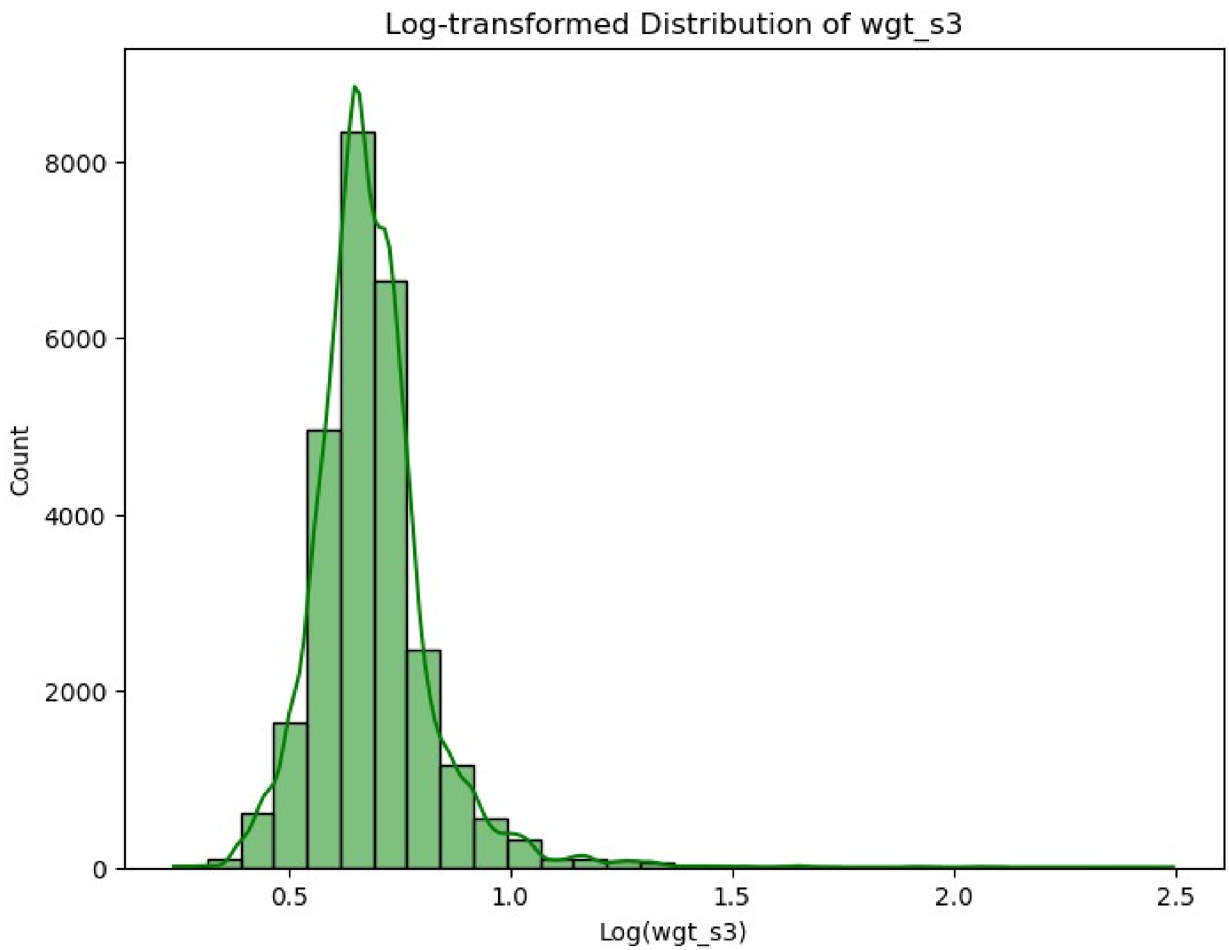
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Load the datasets
data1 = pd.read_csv("C:\\Users\\presc\\OneDrive\\
Fin_wellb05_11_2024.csv")
data2 = pd.read_csv("C:\\Users\\presc\\OneDrive\\
State_Data05_11_2024.csv")

# Clean column names by stripping any leading or trailing spaces
data1.columns = data1.columns.str.strip()
data2.columns = data2.columns.str.strip()

# Select numeric columns from both datasets
numeric_data1 = data1.select_dtypes(include=['float64', 'int64'])
numeric_data2 = data2.select_dtypes(include=['float64', 'int64'])

# Calculate skewness and kurtosis for data1
skewness_data1 = numeric_data1.skew()
kurtosis_data1 = numeric_data1.kurtosis()

# Calculate skewness and kurtosis for data2
```

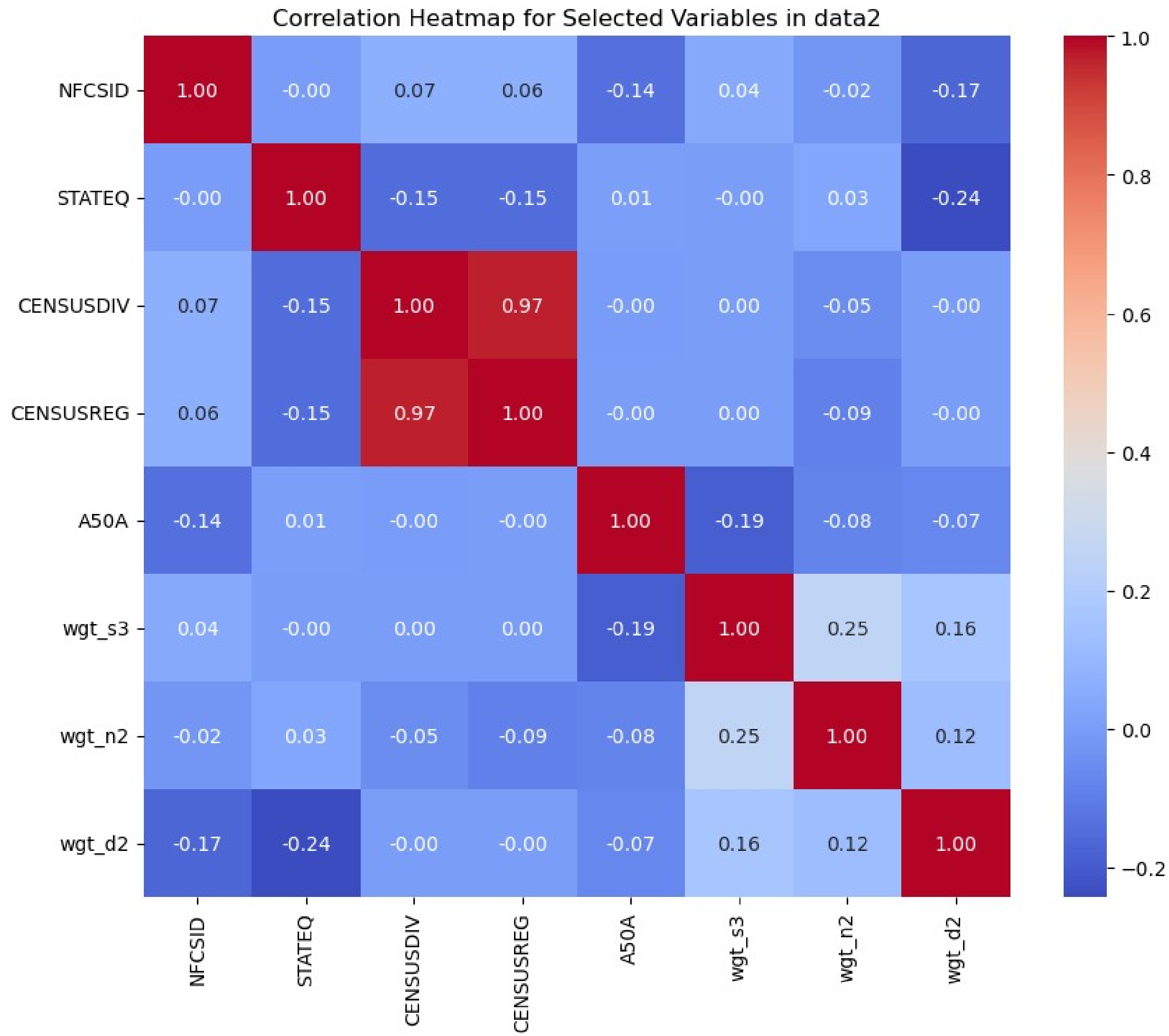
```
skewness_data2 = numeric_data2.skew()
kurtosis_data2 = numeric_data2.kurtosis()

# Visualizations for selected columns from both datasets

# Histogram for 'wgt_s3' in data2 (due to high skewness)
plt.figure(figsize=(8, 6))
sns.histplot(numeric_data2['wgt_s3'], bins=30, kde=True, color='red')
plt.title('Distribution of wgt_s3')
plt.xlabel('wgt_s3')
plt.ylabel('Count')
plt.show()

# Box plot for 'wgt_s3' to visualize outliers
plt.figure(figsize=(6, 4))
sns.boxplot(x=numeric_data2['wgt_s3'], color='orange')
plt.title('Box Plot for wgt_s3')
plt.xlabel('wgt_s3')
plt.show()

# Apply a log transformation for skewed data (e.g., 'wgt_s3') to
normalize
numeric_data2['log_wgt_s3'] = np.log1p(numeric_data2['wgt_s3'])  # Log
transformation (log(1+x))

# Visualize transformed data
plt.figure(figsize=(8, 6))
sns.histplot(numeric_data2['log_wgt_s3'], bins=30, kde=True,
color='green')
plt.title('Log-transformed Distribution of wgt_s3')
plt.xlabel('Log(wgt_s3)')
plt.ylabel('Count')
plt.show()

# Correlation Heatmap for selected numeric columns in data2
plt.figure(figsize=(10, 8))
corr_data2 = numeric_data2[['NFCSID', 'STATEQ', 'CENSUSDIV',
'CENSUSREG', 'A50A', 'wgt_s3', 'wgt_n2', 'wgt_d2']].corr()
sns.heatmap(corr_data2, annot=True, cmap="coolwarm", fmt=".2f")
plt.title('Correlation Heatmap for Selected Variables in data2')
plt.show()

# Display skewness and kurtosis results
print("\nSkewness for numeric columns in data1:")
print(skewness_data1)
print("\nKurtosis for numeric columns in data1:")
print(kurtosis_data1)

print("\nSkewness for numeric columns in data2:")
print(skewness_data2)
```

```python
print("\nKurtosis for numeric columns in data2:")
print(kurtosis_data2)
```




Box Plot for wgt_s3
0
2
4
6
8
10
wgt_s3


Log-transformed Distribution of wgt_s3
Count
0
2000
4000
6000
8000
0.5
1.0
1.5
2.0
2.5
Log(wgt_s3)




```
Skewness for numeric columns in data1:
PUF_ID         -0.031130
sample          1.919704
fpl            -1.694218
SWB_1          -1.203838
SWB_2          -1.297704
                  ...
PPT612          2.205483
PPT1317         2.297279
PPT18OV         0.741445
PCTLT200FPL    -3.012811
finalwt         2.212054
Length: 217, dtype: float64

Kurtosis for numeric columns in data1:
```

```
PUF_ID        -1.116662
sample         2.565448
fpl            1.398116
SWB_1          1.802920
SWB_2          2.250429
                 ...
PPT612         2.865052
PPT1317        3.278518
PPT18OV        0.356909
PCTLT200FPL    8.713753
finalwt        8.852834
Length: 217, dtype: float64

Skewness for numeric columns in data2:
NFCSID       0.000000
STATEQ      -0.008888
CENSUSDIV   -0.122941
CENSUSREG   -0.318593
A50A        -0.161906
               ...
M9           1.460330
M10          0.188802
wgt_n2       1.924560
wgt_d2       1.355259
wgt_s3       5.053243
Length: 86, dtype: float64

Kurtosis for numeric columns in data2:
NFCSID       -1.200000
STATEQ       -1.246895
CENSUSDIV    -1.150638
CENSUSREG    -1.123391
A50A         -1.973932
               ...
M9            0.132876
M10          -1.964325
wgt_n2        6.154866
wgt_d2        1.910834
wgt_s3       84.837405
Length: 86, dtype: float64
```

```
# Load the data from the uploaded CSV files
data1 = pd.read_csv("C:\\Users\\presc\\OneDrive\\
Fin_wellb05_11_2024.csv")
data2 = pd.read_csv("C:\\Users\\presc\\OneDrive\\
State_Data05_11_2024.csv")

# Clean column names by stripping any leading or trailing spaces
```

```
data1.columns = data1.columns.str.strip()
data2.columns = data2.columns.str.strip()

# Display the first few rows and column names to inspect available
data
data1.head(), data2.head(), data1.columns, data2.columns
```

```
(   PUF_ID  sample  fpl  SWB_1  SWB_2  SWB_3  FWBscore  FWB1_1  FWB1_2
FWB1_3  \
 0   10350       2    3      5      5      6        55       3       3
3
 1    7740       1    3      6      6      6        51       2       2
3
 2   13699       1    3      4      3      4        49       3       3
3
 3    7267       1    3      6      6      6        49       3       3
3
 4    7375       1    3      4      4      4        49       3       3
3

    ...  PPMSACAT  PPREG4  PPREG9  PPT01  PPT25  PPT612  PPT1317
PPT18OV  \
 0  ...         1       4       8      0      0       0        0
1
 1  ...         1       2       3      0      0       0        0
2
 2  ...         1       4       9      0      0       0        1
2
 3  ...         1       3       7      0      0       0        0
1
 4  ...         1       2       4      0      0       1        0
4

    PCTLT200FPL   finalwt
 0            0  0.367292
 1            0  1.327561
 2            1  0.835156
 3            0  1.410871
 4            1  4.260668

 [5 rows x 217 columns],
        NFCSID  STATEQ  CENSUSDIV  CENSUSREG  A50A  A3Ar_w  A50B
A4A_new_w  \
 0  2021010001      41          5          3     2       2     8
1
 1  2021010002      36          3          2     2       2     8
1
 2  2021010003       3          8          4     1       6     6
1
 3  2021010004       3          8          4     2       4    10
```

```
2
 4  2021010005      36          3          2     2       4    10
1

    A5_2015  A6  ...  M6  M7  M8  M31  M50  M9  M10    wgt_n2
wgt_d2  \
 0        6   1  ...   1   3   2    2    2   1    2  0.834316
0.539386
 1        6   4  ...  98  98  98   98   98   1   98  1.083618
1.075806
 2        6   4  ...   1   3  98    3    1   1    2  0.396368
2.123406
 3        2   1  ...   1  98  98    1    1   1    2  0.374328
2.372112
 4        3   4  ...   1  98  98   98   98  98   98  1.362034
1.159651

      wgt_s3
 0  0.725252
 1  0.930410
 2  0.944175
 3  1.011643
 4  0.907194

 [5 rows x 126 columns],
 Index(['PUF_ID', 'sample', 'fpl', 'SWB_1', 'SWB_2', 'SWB_3',
'FWBscore',
        'FWB1_1', 'FWB1_2', 'FWB1_3',
        ...
        'PPMSACAT', 'PPREG4', 'PPREG9', 'PPT01', 'PPT25', 'PPT612',
'PPT1317',
        'PPT18OV', 'PCTLT200FPL', 'finalwt'],
       dtype='object', length=217),
 Index(['NFCSID', 'STATEQ', 'CENSUSDIV', 'CENSUSREG', 'A50A',
'A3Ar_w', 'A50B',
        'A4A_new_w', 'A5_2015', 'A6',
        ...
        'M6', 'M7', 'M8', 'M31', 'M50', 'M9', 'M10', 'wgt_n2',
'wgt_d2',
        'wgt_s3'],
       dtype='object', length=126))

# Choose 'wgt_s3' for correlation calculation
correlations = data2_numeric.corr()['wgt_s3'][['wgt_n2', 'wgt_d2',
'M6', 'M7', 'M8']]

# Display the correlations
correlations
```

```
wgt_n2    0.254946
wgt_d2    0.161553
M6        0.037669
M7        0.025827
M8        0.000171
Name: wgt_s3, dtype: float64
```

```
# Convert all columns to numeric in data1, forcing errors to NaN for
non-numeric data
data1_numeric = data1.apply(pd.to_numeric, errors='coerce')

# Choose a relevant column for correlation calculation (e.g., 'SWB_1'
or 'FWBscore')
# I'll calculate the correlation for 'FWBscore' in data1 with other
columns
correlations_data1 = data1_numeric.corr()['FWBscore'][['SWB_1',
'SWB_2', 'SWB_3', 'FWB1_1', 'FWB1_2', 'FWB1_3']]

# Display the correlations
correlations_data1
```

```
SWB_1     0.476626
SWB_2     0.338584
SWB_3     0.194649
FWB1_1    0.702372
FWB1_2    0.663491
FWB1_3   -0.767789
Name: FWBscore, dtype: float64
```

```
from scipy.stats import ttest_ind

# Create two groups based on the median of 'wgt_s3'
median_wgt_s3 = data2['wgt_s3'].median()
high_wgt_s3 = data2[data2['wgt_s3'] > median_wgt_s3]['wgt_n2']
low_wgt_s3 = data2[data2['wgt_s3'] <= median_wgt_s3]['wgt_n2']

# Perform t-test
t_stats, p_value = ttest_ind(high_wgt_s3, low_wgt_s3)
t_stats, p_value
```

```
(41.35274727648881, 0.0)
```

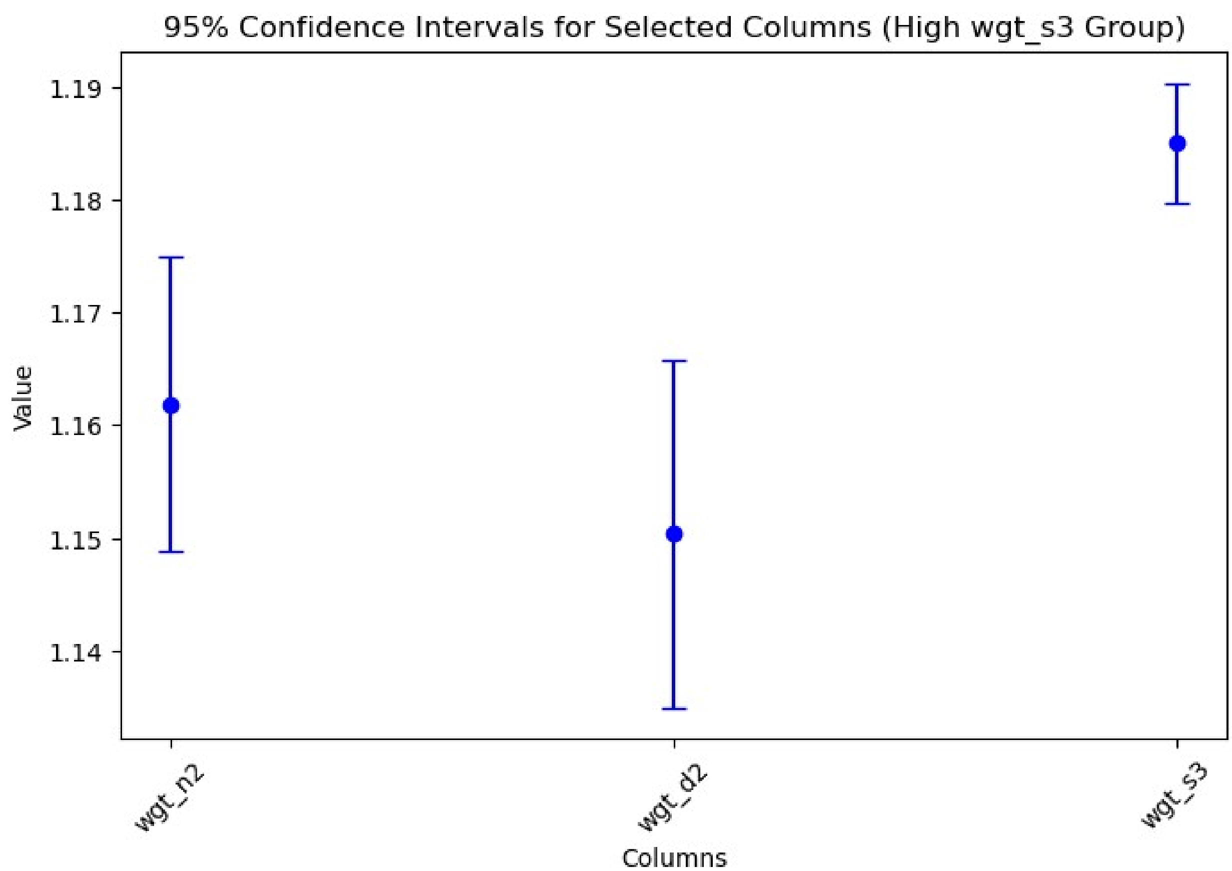
```
from scipy.stats import ttest_ind
```

```python
# Create two groups based on the median of 'FWBscore'
median_fwbscore = data1['FWBscore'].median()
high_fwbscore = data1[data1['FWBscore'] > median_fwbscore][['SWB_1',
'SWB_2', 'SWB_3']]
low_fwbscore = data1[data1['FWBscore'] <= median_fwbscore][['SWB_1',
'SWB_2', 'SWB_3']]

# Perform t-test between the groups for each of the SWB columns
t_stats_swb1, p_value_swb1 = ttest_ind(high_fwbscore['SWB_1'],
low_fwbscore['SWB_1'])
t_stats_swb2, p_value_swb2 = ttest_ind(high_fwbscore['SWB_2'],
low_fwbscore['SWB_2'])
t_stats_swb3, p_value_swb3 = ttest_ind(high_fwbscore['SWB_3'],
low_fwbscore['SWB_3'])

# Display the t-test results
(t_stats_swb1, p_value_swb1), (t_stats_swb2, p_value_swb2),
(t_stats_swb3, p_value_swb3)
```

```
((32.56897189689999, 2.0687076763749525e-215),
 (22.277291834285066, 6.065925644079718e-106),
 (12.64494504725989, 3.229794386642794e-36))
```

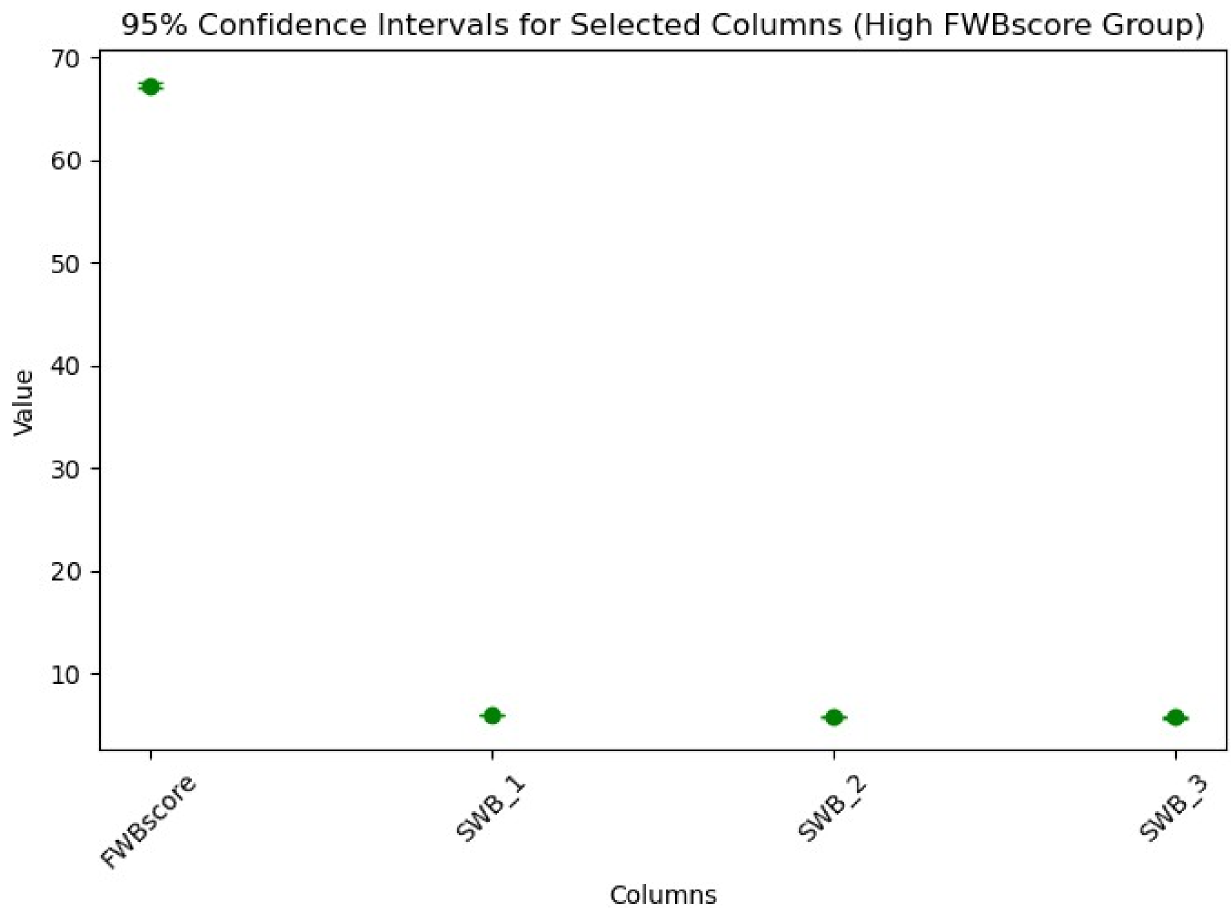
```python
# Re-run the analysis and visualization for the confidence intervals

import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt

# List of columns to analyze
columns_to_analyze = ['wgt_n2', 'wgt_d2', 'wgt_s3']

# Initialize a dictionary to store confidence intervals for each
column
conf_intervals = {}

# Calculate the confidence intervals for each column
for col in columns_to_analyze:
    # Extract the high 'wgt_s3' group for each column
    high_wgt_s3_group = data2[data2['wgt_s3'] > median_wgt_s3][col]

    # Calculate the mean and standard error of the mean (SEM)
    mean_value = np.mean(high_wgt_s3_group)
    sem_value = stats.sem(high_wgt_s3_group)

    # 95% Confidence Interval
    conf_int = stats.t.interval(0.95, len(high_wgt_s3_group)-1,
loc=mean_value, scale=sem_value)
    conf_intervals[col] = conf_int
```

```python
# Visualize the confidence intervals
fig, ax = plt.subplots(figsize=(8, 5))
columns = list(conf_intervals.keys())
lower_bounds = [conf_intervals[col][0] for col in columns]
upper_bounds = [conf_intervals[col][1] for col in columns]
means = [np.mean(data2[data2['wgt_s3'] > median_wgt_s3][col]) for col
in columns]

ax.errorbar(columns, means, yerr=[np.array(means) -
np.array(lower_bounds), np.array(upper_bounds) - np.array(means)],
            fmt='o', color='b', capsize=5)

ax.set_title("95% Confidence Intervals for Selected Columns (High
wgt_s3 Group)")
ax.set_ylabel("Value")
ax.set_xlabel("Columns")

plt.xticks(rotation=45)
plt.show()

conf_intervals  # Display the calculated confidence intervals for each
column
```



```
{'wgt_n2': (1.1487908538517602, 1.1749385263267322),
 'wgt_d2': (1.1350079390435543, 1.1657435324609446),
 'wgt_s3': (1.1796633747356882, 1.1903565932367266)}
```

```python
# Re-running the analysis for data1 to calculate the confidence
intervals
columns_to_analyze_data1 = ['FWBscore', 'SWB_1', 'SWB_2', 'SWB_3']

# Initialize a dictionary to store confidence intervals for each
column
conf_intervals_data1 = {}

# Calculate the confidence intervals for each column in data1
for col in columns_to_analyze_data1:
    # Extract the high 'FWBscore' group for each column
    median_fwbscore = data1['FWBscore'].median()
    high_fwbscore_group = data1[data1['FWBscore'] > median_fwbscore]
[col]

    # Calculate the mean and standard error of the mean (SEM)
    mean_value = np.mean(high_fwbscore_group)
    sem_value = stats.sem(high_fwbscore_group)

    # 95% Confidence Interval
    conf_int = stats.t.interval(0.95, len(high_fwbscore_group)-1,
loc=mean_value, scale=sem_value)
    conf_intervals_data1[col] = conf_int

# Visualize the confidence intervals for data1
fig, ax = plt.subplots(figsize=(8, 5))
columns = list(conf_intervals_data1.keys())
lower_bounds = [conf_intervals_data1[col][0] for col in columns]
upper_bounds = [conf_intervals_data1[col][1] for col in columns]
means = [np.mean(data1[data1['FWBscore'] > median_fwbscore][col]) for
col in columns]

ax.errorbar(columns, means, yerr=[np.array(means) -
np.array(lower_bounds), np.array(upper_bounds) - np.array(means)],
            fmt='o', color='g', capsize=5)

ax.set_title("95% Confidence Intervals for Selected Columns (High
FWBscore Group)")
ax.set_ylabel("Value")
ax.set_xlabel("Columns")

plt.xticks(rotation=45)
plt.show()

conf_intervals_data1  # Display the calculated confidence intervals
for each column in data1
```




```
{'FWBscore': (66.96671873130188, 67.58474006959585),
 'SWB_1': (5.893711904545474, 5.974194475704606),
 'SWB_2': (5.740820278973357, 5.833402228240494),
 'SWB_3': (5.637827628562086, 5.743384298337562)}
```

```python
from scipy.stats import f_oneway

# Divide data1 into three groups based on FWBscore (tertiles)
low_fwbscore = data1[data1['FWBscore'] <=
data1['FWBscore'].quantile(1/3)]['SWB_1']
medium_fwbscore = data1[(data1['FWBscore'] >
data1['FWBscore'].quantile(1/3)) & (data1['FWBscore'] <=
data1['FWBscore'].quantile(2/3))]['SWB_1']
high_fwbscore = data1[data1['FWBscore'] >
data1['FWBscore'].quantile(2/3)]['SWB_1']

# Perform ANOVA Test for SWB_1 across the three FWBscore groups
f_stats_fwbscore, p_value_fwbscore = f_oneway(low_fwbscore,
medium_fwbscore, high_fwbscore)
f_stats_fwbscore, p_value_fwbscore
```

```
(709.1010714492888, 7.422653141933526e-279)
```

```
# Re-run the ANOVA test for wgt_n2 across the three wgt_s3 groups with
the correct variable name
f_stats_wgt_s3, p_value_wgt_s3 = f_oneway(low_wgt_s3, medium_wgt_s3,
high_wgt_s3)

# Display the results
f_stats_wgt_s3, p_value_wgt_s3
```

```
(920.2729280480268, 0.0)
```

```
# Import necessary libraries
import pandas as pd
from scipy.stats import chi2_contingency

# Assuming data2 is already loaded and cleaned

# Categorize wgt_s3 into three groups: Low, Medium, High
data2['wgt_s3_category'] = pd.qcut(data2['wgt_s3'], 3, labels=['Low',
'Medium', 'High'])

# Categorize wgt_n2 into three groups: Low, Medium, High
data2['wgt_n2_category'] = pd.qcut(data2['wgt_n2'], 3, labels=['Low',
'Medium', 'High'])

# Create a contingency table comparing wgt_s3 and wgt_n2 categories
crosstab_wgt_s3_n2 = pd.crosstab(data2['wgt_s3_category'],
data2['wgt_n2_category'])

# Perform Chi-square test
chi_wgt_s3_n2, p_value_wgt_s3_n2, dof_wgt_s3_n2, ex_wgt_s3_n2 =
chi2_contingency(crosstab_wgt_s3_n2)

# Print the results for data2
print(f'Chi-square Statistic: {chi_wgt_s3_n2}')
print(f'P-value: {p_value_wgt_s3_n2}')
```

```
Chi-square Statistic: 1648.8426184158664
P-value: 0.0
```

```
# Import necessary libraries
import pandas as pd
from scipy.stats import chi2_contingency

# Assuming data1 is already loaded and cleaned

# Categorize SWB_1 into three groups: Low, Medium, High
data1['swb1_category'] = pd.qcut(data1['SWB_1'], 3, labels=['Low',
'Medium', 'High'])
```

```python
# Categorize FWBscore into three groups: Low, Medium, High
data1['fwbscore_category'] = pd.qcut(data1['FWBscore'], 3,
labels=['Low', 'Medium', 'High'])

# Create a contingency table comparing SWB_1 and FWBscore categories
crosstab_swb_fwb = pd.crosstab(data1['swb1_category'],
data1['fwbscore_category'])

# Perform Chi-square test
chi_swb_fwb, p_value_swb_fwb, dof_swb_fwb, ex_swb_fwb =
chi2_contingency(crosstab_swb_fwb)

# Print the results for data1
print(f'Chi-square Statistic for data1: {chi_swb_fwb}')
print(f'P-value for data1: {p_value_swb_fwb}')
```

```
Chi-square Statistic for data1: 1213.488409388712
P-value for data1: 1.8969714761906545e-261
```