```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

data1 = pd.read_csv("adultdepression.csv")

# Load the dataset

data2 = pd.read_csv("Telehealth.csv")

#Statistical Report
data1.describe()
```

```
              Year    Frequency  Weighted Frequency     Percent  Lower
95% CL  \
count   161.00000   161.000000        1.540000e+02  161.000000
161.000000
mean   2015.00000   429.776398        8.898917e+05   14.789627
11.955280
std       2.00624   390.297867        6.299145e+05    4.589876
3.705456
min    2012.00000    28.000000        9.230900e+04    3.970000
2.000000
25%    2013.00000   186.000000        4.597088e+05   11.850000
9.650000
50%    2015.00000   314.000000        7.164805e+05   14.520000
11.550000
75%    2017.00000   511.000000        1.109084e+06   17.190000
14.600000
max    2018.00000  1964.000000        3.301418e+06   33.090000
24.600000

       Upper 95% CL
count    161.000000
mean      17.624224
std        5.890040
min        5.340000
25%       13.870000
50%       16.930000
75%       20.050000
max       44.950000
```

```python
data2.describe()
```

```
        TM_Bene_Cnt  Tot_Bene_Cnt   TM_Bene_Pct
count  1.406800e+04  1.407400e+04  14062.000000
mean   1.725387e+05  7.284492e+05      0.277358
std    8.708255e+05  3.192932e+06      0.152943
min    0.000000e+00  1.100000e+01      0.000000
25%    4.318250e+03  1.827300e+04      0.161368
50%    1.989850e+04  8.740950e+04      0.236994
75%    6.966525e+04  2.860108e+05      0.358173
max    2.825518e+07  5.345083e+07      0.813953
```

```
import pandas as pd

# Load data from a CSV file (replace 'your_file.csv' with your file
path)
data1 = pd.read_csv("adultdepression.csv")

# Rename columns
data1 = data1.rename(columns={
    'Year': 'Year',
    'Frequency': 'Frequency',
    'Weighted Frequency': 'Weighted_Frequency',
    'Percent': 'Percent',
    'Lower 95% CL': 'Lower_95_CL',
    'Upper 95% CL': 'Upper_95_CL'
})

# Calculate Mean, Median, and Mode for 'Year' and 'Frequency'
print("\nMean for 'Year':")
print(data1['Year'].mean())
print("\nMedian for 'Year':")
print(data1['Year'].median())
print("\nMode for 'Year':")
print(data1['Year'].mode())

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

```
Mean for 'Year':
2015.0

Median for 'Year':
2015.0

Mode for 'Year':
0    2012
```

```
1    2013
2    2014
3    2015
4    2016
5    2017
6    2018
Name: Year, dtype: int64

Mean for 'Frequency':
429.77639751552795

Median for 'Frequency':
314.0

Mode for 'Frequency':
0    155
Name: Frequency, dtype: int64
```

```python
import pandas as pd

# Load data from a CSV file (replace 'your_file.csv' with your file
path)
data2 = pd.read_csv("Telehealth.csv")

# Rename columns as you specified
data2 = data2.rename(columns={
    'TM_Bene_Cnt': 'TM_Benefit_Count',
    'Tot_Bene_Cnt': 'Total_Benefit_Count',
    'TM_Bene_Pct': 'TM_Benefit_Percentage'
})

# Calculate Mean, Median, and Mode for the renamed columns

print("\nMean for 'TM_Benefit_Count':")
print(data2['TM_Benefit_Count'].mean())

print("\nMedian for 'TM_Benefit_Count':")
print(data2['TM_Benefit_Count'].median())

print("\nMode for 'TM_Benefit_Count':")
print(data2['TM_Benefit_Count'].mode())

print("\nMean for 'Total_Benefit_Count':")
print(data2['Total_Benefit_Count'].mean())

print("\nMedian for 'Total_Benefit_Count':")
print(data2['Total_Benefit_Count'].median())

print("\nMode for 'Total_Benefit_Count':")
print(data2['Total_Benefit_Count'].mode())
```

```python
print("\nMean for 'TM_Benefit_Percentage':")
print(data2['TM_Benefit_Percentage'].mean())

print("\nMedian for 'TM_Benefit_Percentage':")
print(data2['TM_Benefit_Percentage'].median())

print("\nMode for 'TM_Benefit_Percentage':")
print(data2['TM_Benefit_Percentage'].mode())
```

```
Mean for 'TM_Benefit_Count':
172538.72021609326

Median for 'TM_Benefit_Count':
19898.5

Mode for 'TM_Benefit_Count':
0     14.0
1     38.0
2     85.0
3    138.0
Name: TM_Benefit_Count, dtype: float64

Mean for 'Total_Benefit_Count':
728449.1892141538

Median for 'Total_Benefit_Count':
87409.5

Mode for 'Total_Benefit_Count':
0    133.0
1    147.0
2    292.0
3    297.0
4    365.0
5    380.0
6    899.0
Name: Total_Benefit_Count, dtype: float64

Mean for 'TM_Benefit_Percentage':
0.2773583635463803

Median for 'TM_Benefit_Percentage':
0.23699419825

Mode for 'TM_Benefit_Percentage':
0    0.5
Name: TM_Benefit_Percentage, dtype: float64
```

```
import pandas as pd

# Load data from a CSV
data1 = pd.read_csv("adultdepression.csv")
data2 = pd.read_csv("Telehealth.csv")

import pandas as pd

# Load data from CSV files (replace with your file paths if needed)
data1 = pd.read_csv("adultdepression.csv")
data2 = pd.read_csv("Telehealth.csv")

# Ensure relevant columns are numeric for data1 (replace with correct
column names if needed)
data1['Frequency'] = pd.to_numeric(data1['Frequency'],
errors='coerce')
data1['Weighted Frequency'] = pd.to_numeric(data1['Weighted
Frequency'], errors='coerce')
data1['Percent'] = pd.to_numeric(data1['Percent'], errors='coerce')

# Measure of Dispersion for data1

# Convert all columns to numeric, non-numeric values will be set as
NaN
data1_numeric = data1.apply(pd.to_numeric, errors='coerce')

# Compute variance, ignoring NaN values
print("Variance for data1:")
print(data1_numeric.var())

# Compute standard deviation, ignoring NaN values
print("\nStandard Deviation for data1:")
print(data1_numeric.std())

# Ensure relevant columns are numeric using the correct column names
data2['TM_Bene_Cnt'] = pd.to_numeric(data2['TM_Bene_Cnt'],
errors='coerce')
data2['Tot_Bene_Cnt'] = pd.to_numeric(data2['Tot_Bene_Cnt'],
errors='coerce')
data2['TM_Bene_Pct'] = pd.to_numeric(data2['TM_Bene_Pct'],
errors='coerce')

# Select only numeric columns for data2
data2_numeric = data2.select_dtypes(include=[np.number])

# Calculate the variance and standard deviation for numeric columns
print("\nVariance for data2:")
print(data2_numeric.var())

print("\nStandard Deviation for data2:")
```

```
print(data2_numeric.std())
```

```
Variance for data1:
Year                   4.025000e+00
Strata                          NaN
Strata Name                     NaN
Frequency              1.523324e+05
Weighted Frequency     3.967923e+11
Percent                2.106696e+01
Lower 95% CL           1.373040e+01
Upper 95% CL           3.469257e+01
dtype: float64

Standard Deviation for data1:
Year                          2.006240
Strata                             NaN
Strata Name                        NaN
Frequency                  390.297867
Weighted Frequency    629914.504700
Percent                       4.589876
Lower 95% CL                  3.705456
Upper 95% CL                  5.890040
dtype: float64

Variance for data2:
TM_Bene_Cnt     7.583370e+11
Tot_Bene_Cnt    1.019482e+13
TM_Bene_Pct     2.339146e-02
dtype: float64

Standard Deviation for data2:
TM_Bene_Cnt     8.708255e+05
Tot_Bene_Cnt    3.192932e+06
TM_Bene_Pct     1.529427e-01
dtype: float64
```

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

# Skewness
print("\nSkewness:  ")
print(numeric_data.skew())

# Kurtosis
print("\nKurtosis:  ")
print(numeric_data.kurtosis())
```

```
Skewness:
Year                   0.000000
```

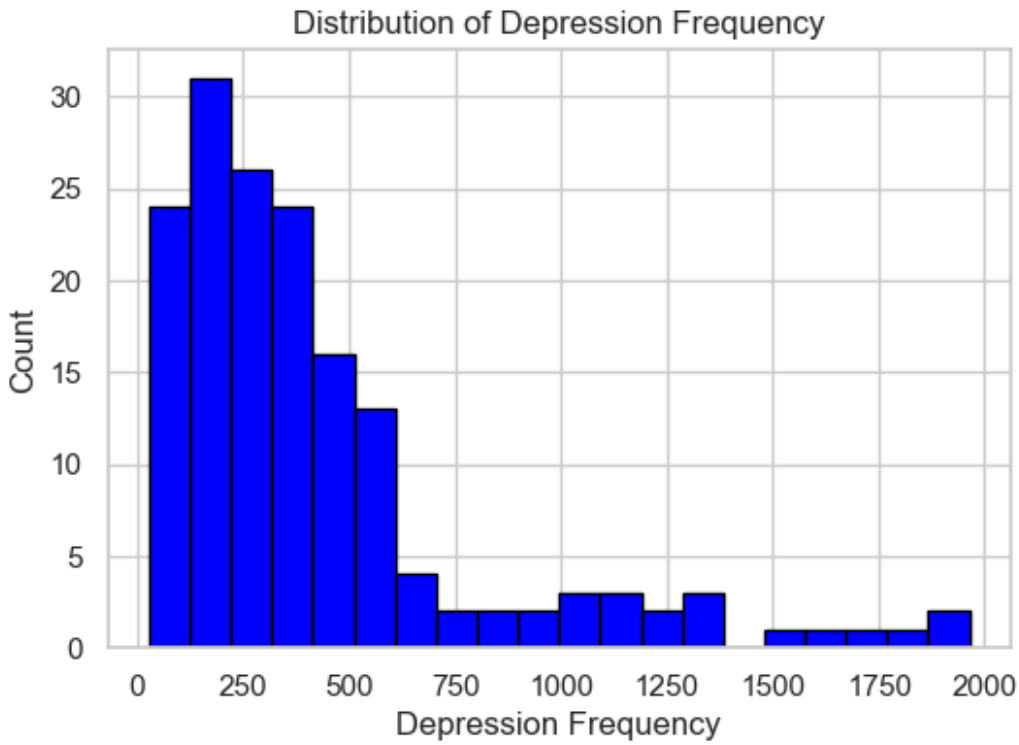
```
Frequency              1.987604
Weighted Frequency     1.399177
Percent                0.696316
Lower 95% CL           0.216567
Upper 95% CL           1.174118
dtype: float64

Kurtosis:
Year                  -1.251493
Frequency              4.037654
Weighted Frequency     1.931348
Percent                1.698155
Lower 95% CL           0.814878
Upper 95% CL           3.017912
dtype: float64
```

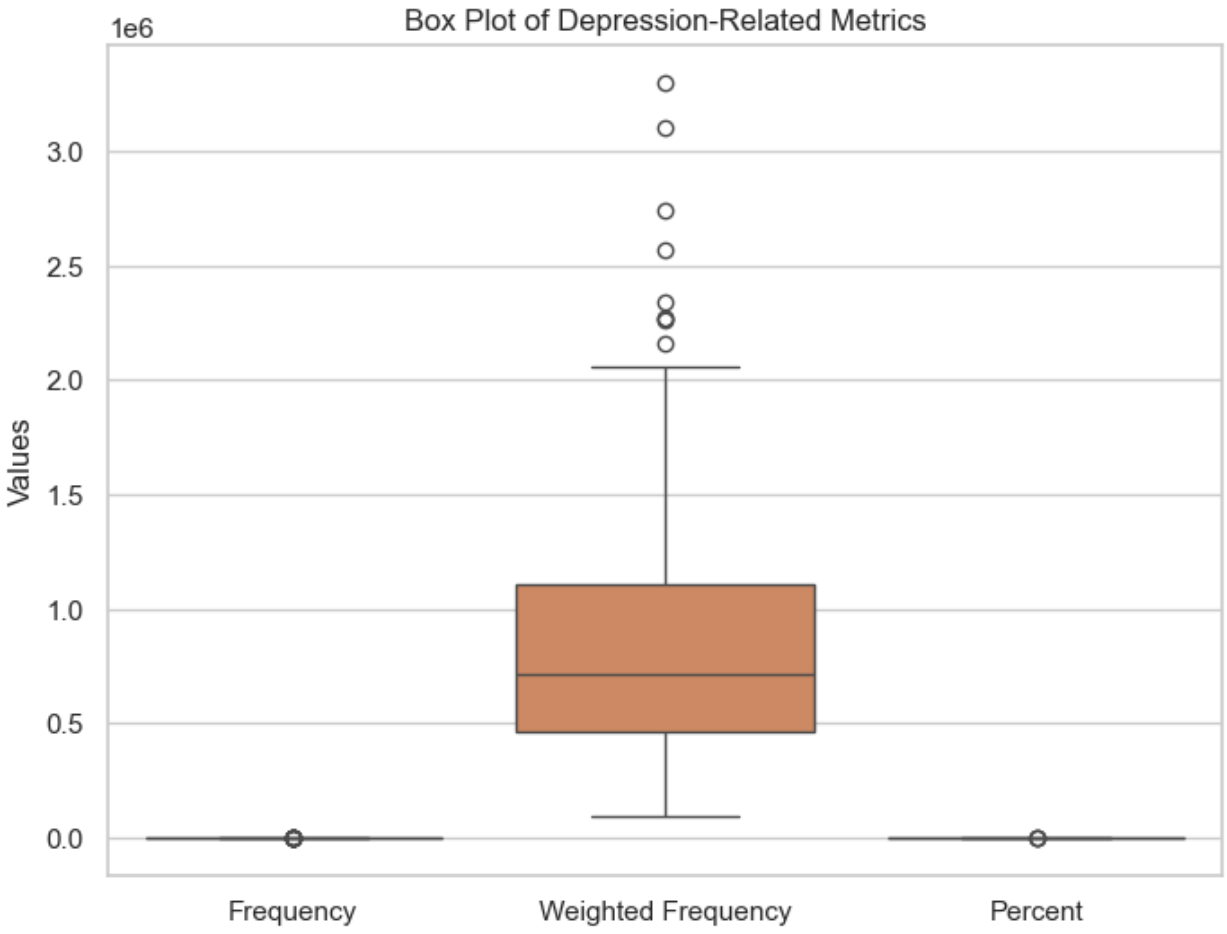
```
# Select only numeric columns from data2
numeric_data2 = data2.select_dtypes(include=['float64', 'int64'])

# Skewness
print("\nSkewness for data2:  ")
print(numeric_data2.skew())

# Kurtosis
print("\nKurtosis for data2:  ")
print(numeric_data2.kurtosis())
```

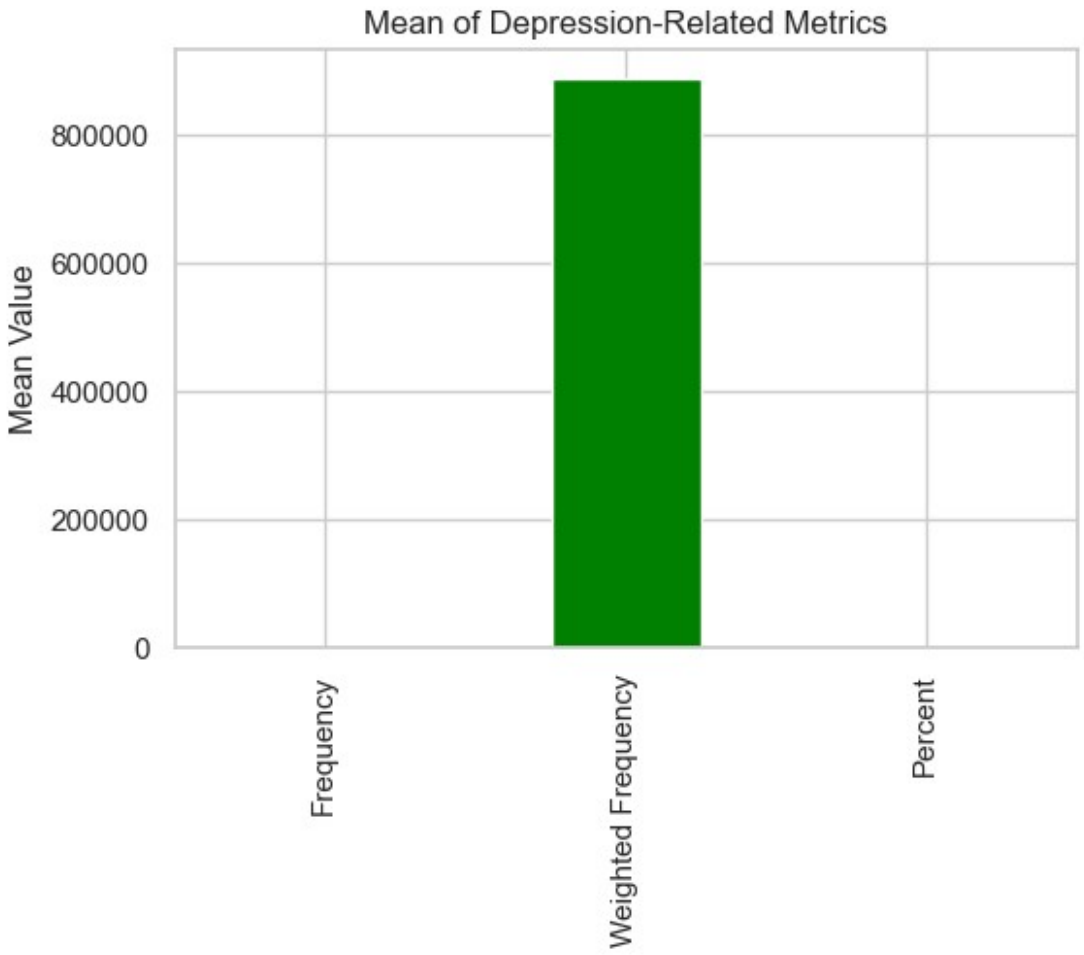
```
Skewness for data2:
TM_Bene_Cnt     15.825941
Tot_Bene_Cnt     8.311630
TM_Bene_Pct      0.957053
dtype: float64

Kurtosis for data2:
TM_Bene_Cnt     359.051867
Tot_Bene_Cnt     85.812126
TM_Bene_Pct       0.244655
dtype: float64
```

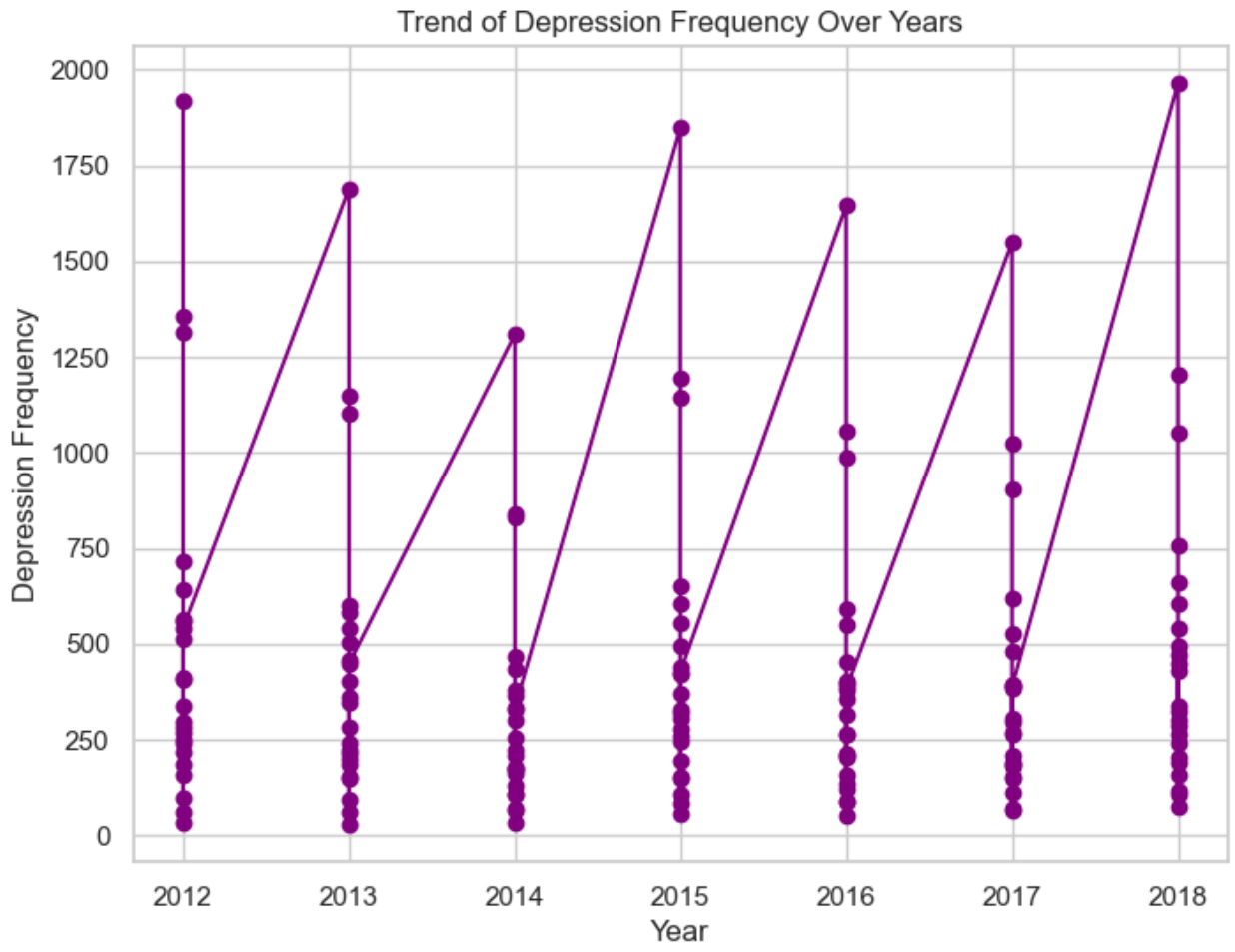
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Set a style for the plots
sns.set(style="whitegrid")

# 1. Histogram for 'Frequency' (distribution of depression frequency)
plt.figure(figsize=(6, 4))
plt.hist(data1['Frequency'], bins=20, color='blue', edgecolor='black')
plt.xlabel('Depression Frequency')
```

```
plt.ylabel('Count')
plt.title('Distribution of Depression Frequency')
plt.show()

# 2. Box Plot for 'Frequency', 'Weighted Frequency', and 'Percent'
plt.figure(figsize=(8, 6))
sns.boxplot(data=data1[['Frequency', 'Weighted Frequency',
'Percent']])
plt.title('Box Plot of Depression-Related Metrics')
plt.ylabel('Values')
plt.show()

# 3. Bar Plot for Mean Values of 'Frequency', 'Weighted Frequency',
and 'Percent'
mean_values = data1[['Frequency', 'Weighted Frequency',
'Percent']].mean()
plt.figure(figsize=(6, 4))
mean_values.plot(kind='bar', color='green')
plt.ylabel('Mean Value')
plt.title('Mean of Depression-Related Metrics')
plt.show()

# 4. Line Plot of 'Year' vs. 'Frequency' (Trend over years)
plt.figure(figsize=(8, 6))
plt.plot(data1['Year'], data1['Frequency'], marker='o',
color='purple')
plt.xlabel('Year')
plt.ylabel('Depression Frequency')
plt.title('Trend of Depression Frequency Over Years')
plt.show()
```


Distribution of Depression Frequency
Count
0
5
10
15
20
25
30
0
250
500
750
1000
1250
1500
1750
2000
Depression Frequency

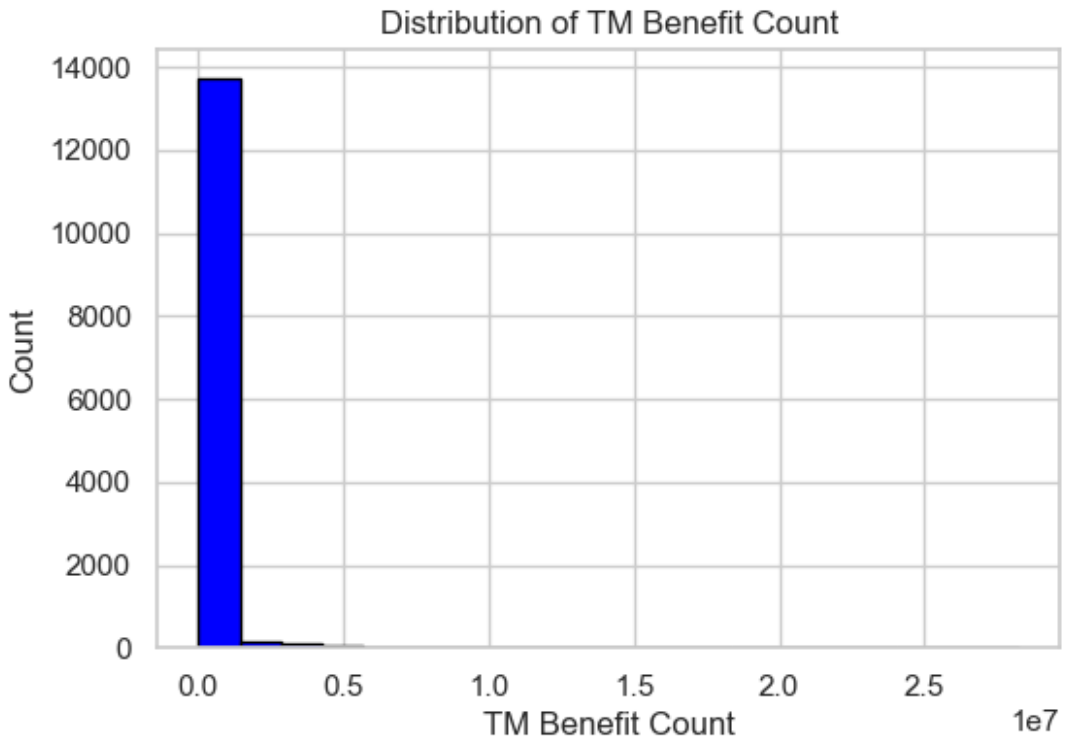

Box Plot of Depression-Related Metrics
1e6
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Values
Frequency
Weighted Frequency
Percent

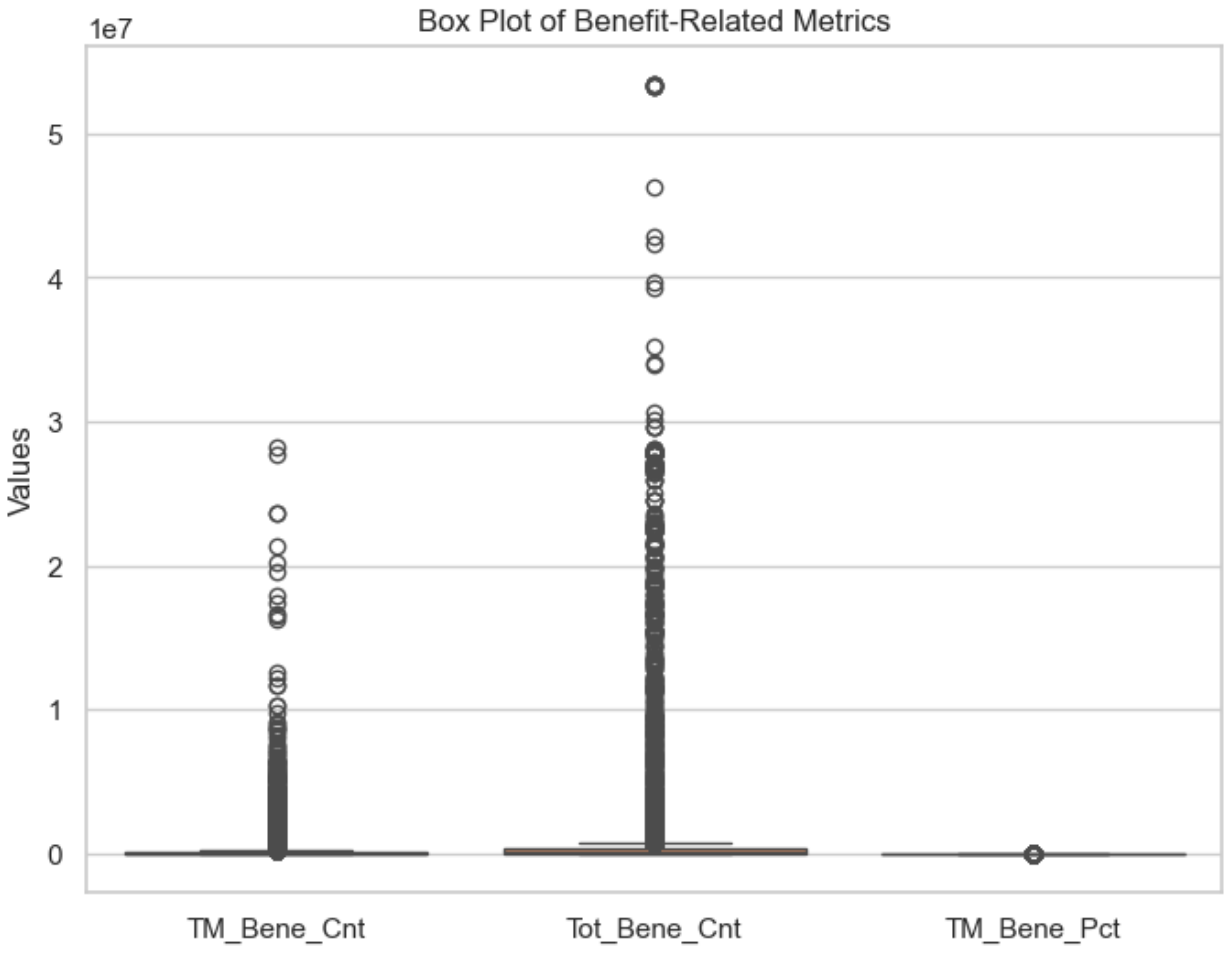

Mean of Depression-Related Metrics
Mean Value
800000
600000
400000
200000
0
Frequency
Weighted Frequency
Percent

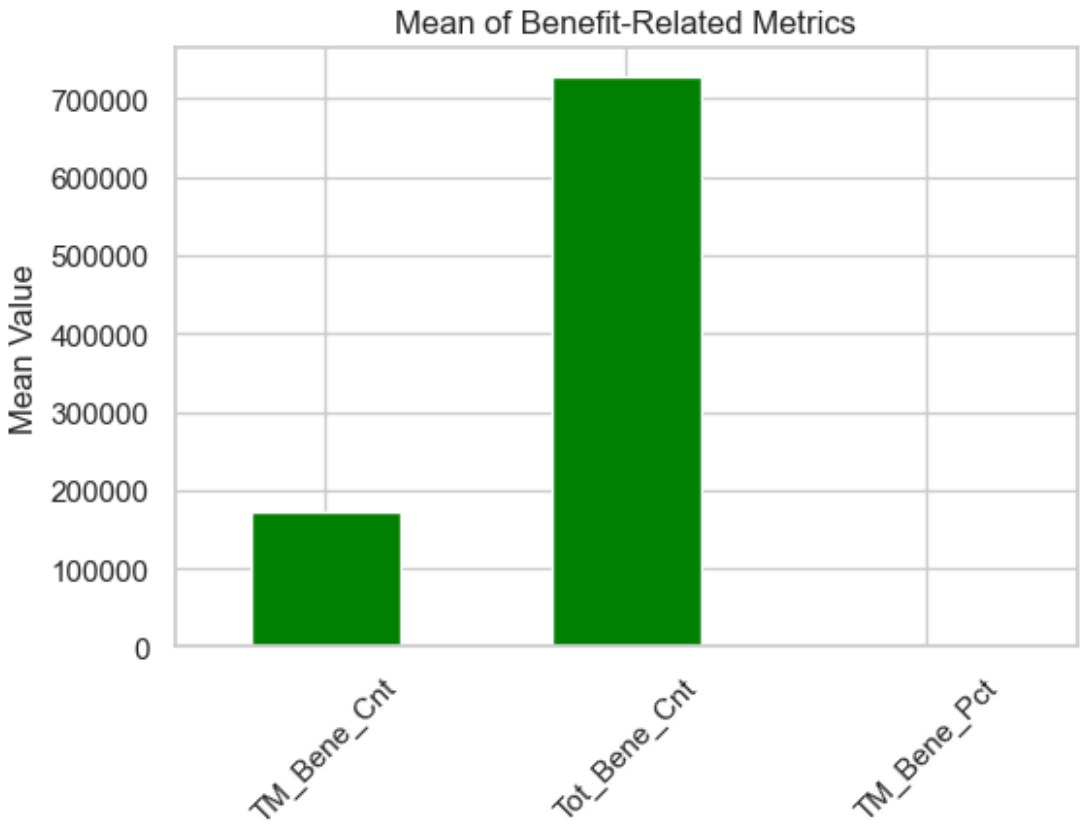


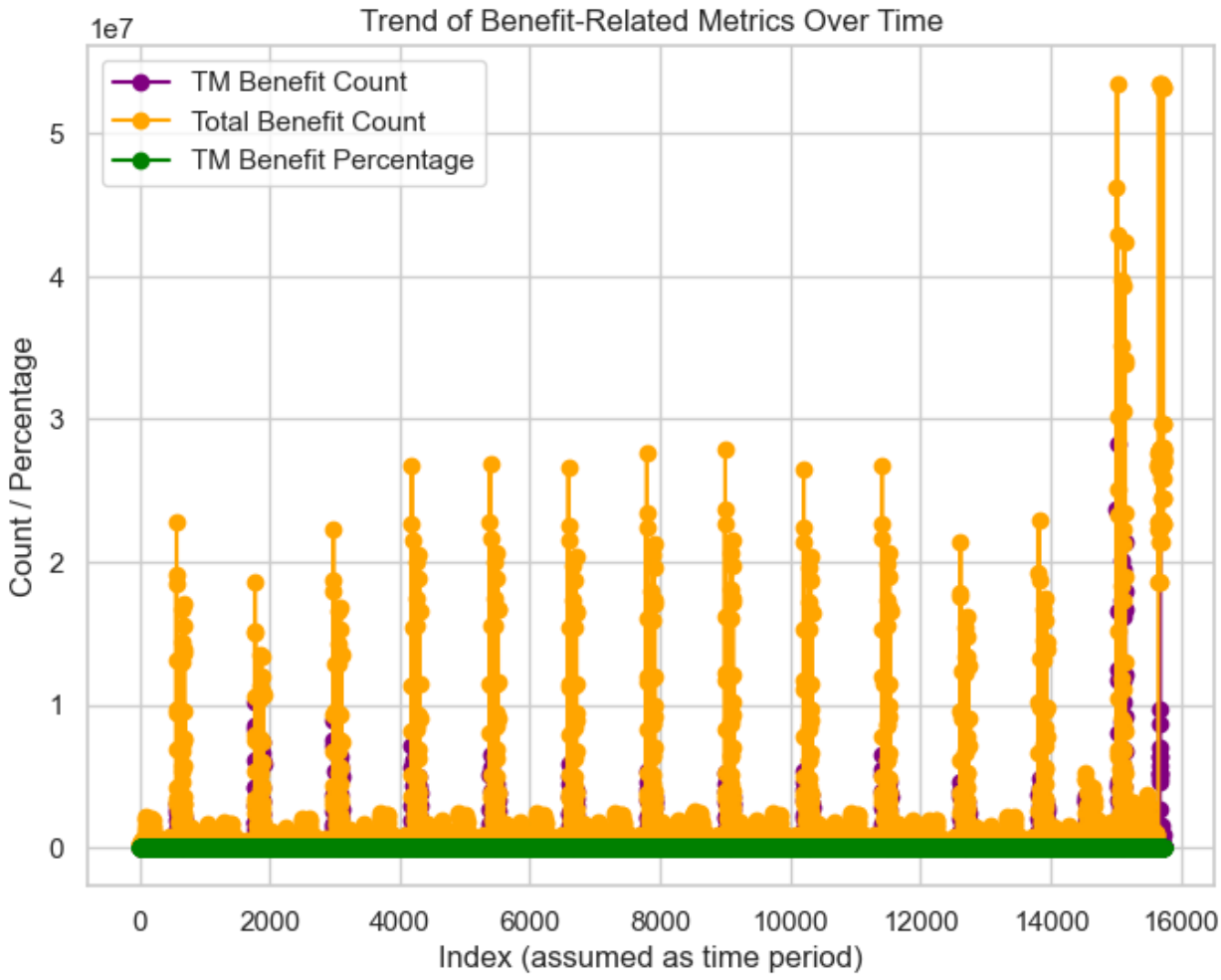

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Set a style for the plots
sns.set(style="whitegrid")

# 1. Histogram for 'TM_Bene_Cnt' (distribution of TM Benefit Count)
plt.figure(figsize=(6, 4))
plt.hist(data2['TM_Bene_Cnt'], bins=20, color='blue',
edgecolor='black')
plt.xlabel('TM Benefit Count')
plt.ylabel('Count')
plt.title('Distribution of TM Benefit Count')
plt.show()

# 2. Box Plot for 'TM_Bene_Cnt', 'Tot_Bene_Cnt', and 'TM_Bene_Pct'
plt.figure(figsize=(8, 6))
sns.boxplot(data=data2[['TM_Bene_Cnt', 'Tot_Bene_Cnt',
'TM_Bene_Pct']])
```

```
plt.title('Box Plot of Benefit-Related Metrics')
plt.ylabel('Values')
plt.show()

# 3. Bar Plot for Mean Values of 'TM_Bene_Cnt', 'Tot_Bene_Cnt', and
'TM_Bene_Pct'
mean_values = data2[['TM_Bene_Cnt', 'Tot_Bene_Cnt',
'TM_Bene_Pct']].mean()
plt.figure(figsize=(6, 4))
mean_values.plot(kind='bar', color='green')
plt.ylabel('Mean Value')
plt.title('Mean of Benefit-Related Metrics')
plt.xticks(rotation=45)
plt.show()

# 4. Line Plot for all metrics over their index (assuming each row
represents a year or a time period)
plt.figure(figsize=(8, 6))
plt.plot(data2.index, data2['TM_Bene_Cnt'], marker='o',
color='purple', label='TM Benefit Count')
plt.plot(data2.index, data2['Tot_Bene_Cnt'], marker='o',
color='orange', label='Total Benefit Count')
plt.plot(data2.index, data2['TM_Bene_Pct'], marker='o', color='green',
label='TM Benefit Percentage')
plt.xlabel('Index (assumed as time period)')
plt.ylabel('Count / Percentage')
plt.title('Trend of Benefit-Related Metrics Over Time')
plt.legend()
plt.show()
```


Distribution of TM Benefit Count
14000
12000
10000
8000
6000
4000
2000
0
Count
0.0
0.5
1.0
1.5
2.0
2.5
TM Benefit Count
1e7


Box Plot of Benefit-Related Metrics
1e7
5
4
3
2
1
0
Values
TM_Bene_Cnt
Tot_Bene_Cnt
TM_Bene_Pct


Mean of Benefit-Related Metrics
Mean Value
700000
600000
500000
400000
300000
200000
100000
0
TM_Bene_Cnt
Tot_Bene_Cnt
TM_Bene_Pct




```
# Convert columns to numeric, forcing errors to NaN for non-numeric
data
data2_numeric = data2.apply(pd.to_numeric, errors='coerce')

# Calculate correlation between TM_Bene_Cnt and other columns
correlations = data2_numeric.corr()['TM_Bene_Cnt'][['Tot_Bene_Cnt',
'TM_Bene_Pct']]

# Display the correlations
print(correlations)
```

```
Tot_Bene_Cnt    0.697716
TM_Bene_Pct     0.111033
Name: TM_Bene_Cnt, dtype: float64
```

```
# Drop non-numeric columns from data1
data1_numeric = data1.select_dtypes(include=[float, int])

# Calculate correlation between Year and other numeric columns
```

```
correlations = data1_numeric.corr()['Year'][['Frequency', 'Weighted
Frequency', 'Percent', 'Lower 95% CL', 'Upper 95% CL']]

# Display the correlations
print(correlations)
```

```
Frequency            -0.003081
Weighted Frequency    0.205127
Percent               0.453173
Lower 95% CL          0.317585
Upper 95% CL          0.506528
Name: Year, dtype: float64
```

```
from scipy.stats import ttest_ind

# Create two groups: High TM_Bene_Cnt and Low TM_Bene_Cnt based on the
median
median_bene_cnt = data2['TM_Bene_Cnt'].median()
high_bene_cnt = data2[data2['TM_Bene_Cnt'] > median_bene_cnt]
['Tot_Bene_Cnt']
low_bene_cnt = data2[data2['TM_Bene_Cnt'] <= median_bene_cnt]
['Tot_Bene_Cnt']

# Perform t-test
t_stats, p_value = ttest_ind(high_bene_cnt, low_bene_cnt)
print(f'T-Statistics: {t_stats}, p-value: {p_value}')
```

```
T-Statistics: nan, p-value: nan
```

```
from scipy.stats import ttest_ind

# Create two groups: High Year and Low Year based on the median
median_year = data1['Year'].median()
high_year = data1[data1['Year'] > median_year]['Frequency']
low_year = data1[data1['Year'] <= median_year]['Frequency']

# Perform t-test
t_stats, p_value = ttest_ind(high_year, low_year)
print(f'T-Statistics: {t_stats}, p-value: {p_value}')
```

```
T-Statistics: 0.008716260375535507, p-value: 0.9930564450859358
```

```
import numpy as np
import scipy.stats as stats

# Define the high TM_Bene_Cnt group based on median
median_bene_cnt = data2['TM_Bene_Cnt'].median()
high_bene_cnt_totals = data2[data2['TM_Bene_Cnt'] > median_bene_cnt]
['Tot_Bene_Cnt']

# Calculate mean and standard error of the mean (SEM)
```

```
mean_bene_totals = np.mean(high_bene_cnt_totals)
sem_bene_totals = stats.sem(high_bene_cnt_totals)

# 95% Confidence Interval for Tot_Bene_Cnt in the high TM_Bene_Cnt
group
conf_int_bene_totals = stats.t.interval(0.95,
len(high_bene_cnt_totals)-1, loc=mean_bene_totals,
scale=sem_bene_totals)
conf_int_bene_totals
```

```
(1256867.3028531452, 1451537.826234145)
```

```
import numpy as np
import scipy.stats as stats

# Define the high Year group based on median
median_year = data1['Year'].median()
high_year_frequency = data1[data1['Year'] > median_year]['Frequency']

# Calculate mean and standard error of the mean (SEM)
mean_frequency = np.mean(high_year_frequency)
sem_frequency = stats.sem(high_year_frequency)

# 95% Confidence Interval for Frequency in the high Year group
conf_int_frequency = stats.t.interval(0.95, len(high_year_frequency)-
1, loc=mean_frequency, scale=sem_frequency)
conf_int_frequency
```

```
(338.05468344489464, 522.1192295985836)
```

```
from scipy.stats import f_oneway

# Divide data1 into three groups based on Year (tertiles)
low_year = data1[data1['Year'] <= data1['Year'].quantile(1/3)]
['Frequency']
medium_year = data1[(data1['Year'] > data1['Year'].quantile(1/3)) &
(data1['Year'] <= data1['Year'].quantile(2/3))]['Frequency']
high_year = data1[data1['Year'] > data1['Year'].quantile(2/3)]
['Frequency']

# Perform ANOVA Test for Frequency across the three Year groups
f_stats_year, p_value_year = f_oneway(low_year, medium_year,
high_year)
print(f'F-Statistics for Year: {f_stats_year}, p-value:
{p_value_year}')
```

```
F-Statistics for Year: 0.05502749786410392, p-value:
0.9464772523690465
```

```
from scipy.stats import f_oneway
```

```
# Divide data2 into three groups based on TM_Bene_Pct (tertiles)
low_bene_pct = data2[data2['TM_Bene_Pct'] <=
data2['TM_Bene_Pct'].quantile(1/3)]['Tot_Bene_Cnt']
medium_bene_pct = data2[(data2['TM_Bene_Pct'] >
data2['TM_Bene_Pct'].quantile(1/3)) & (data2['TM_Bene_Pct'] <=
data2['TM_Bene_Pct'].quantile(2/3))]['Tot_Bene_Cnt']
high_bene_pct = data2[data2['TM_Bene_Pct'] >
data2['TM_Bene_Pct'].quantile(2/3)]['Tot_Bene_Cnt']

# Perform ANOVA Test for Tot_Bene_Cnt across the three TM_Bene_Pct
groups
f_stats_bene_pct, p_value_bene_pct = f_oneway(low_bene_pct,
medium_bene_pct, high_bene_pct)
print(f'F-Statistics for TM_Bene_Pct: {f_stats_bene_pct}, p-value:
{p_value_bene_pct}')
```

```
F-Statistics for TM_Bene_Pct: 8.00339036906897, p-value:
0.00033585275576006904
```

```
from scipy.stats import chi2_contingency
import pandas as pd

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

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

# Create a contingency table comparing Frequency and Percent
categories
crosstab1 = pd.crosstab(data1['frequency_category'],
data1['percent_category'])

# Perform Chi-square test
chi1, p_value1, dof1, ex1 = chi2_contingency(crosstab1)

# Print the results for data1
print(f'Chi-square for data1: {chi1}, p-value: {p_value1}')
```

```
Chi-square for data1: 5.114783903004305, p-value: 0.2757214193972128
```

```
from scipy.stats import chi2_contingency
import pandas as pd

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

# Categorize TM_Bene_Pct into three groups: Low, Medium, High
```

```
data2['bene_pct_category'] = pd.qcut(data2['TM_Bene_Pct'], 3,
labels=['Low', 'Medium', 'High'])

# Create a contingency table comparing TM_Bene_Cnt and TM_Bene_Pct
categories
crosstab2 = pd.crosstab(data2['bene_count_category'],
data2['bene_pct_category'])

# Perform Chi-square test
chi2, p_value2, dof2, ex2 = chi2_contingency(crosstab2)

# Print the results for data2
print(f'Chi-square for data2: {chi2}, p-value: {p_value2}')
```

```
Chi-square for data2: 594.388394493675, p-value: 2.5392138018373606e-
127
```