mental-health-ids-1

April 9, 2025

1 MENTAL HEALTH DATASET

1.1 INTRODUCTION

This dataset focuses on analyzing various factors that influence mental health among individuals, particularly students. It includes both subjective and objective stress indicators such as:

Survey-Based Stress Scores (Survey_Stress_Score)

Wearable Device-Based Stress Scores (Wearable_Stress_Score)

Lifestyle Factors like Sleep_Hours and Screen_Time_Hours

Support Systems indicating emotional or mental support availability

Demographic Information such as Gender, Academic_Level, and Country

The goal of this dataset is to explore the relationships between stress levels and lifestyle or support-related variables, helping to identify trends, correlations, and possible areas of intervention for mental well-being.

2 DATA PROCESSING

Data preprocessing is a critical step in any data analysis or machine learning pipeline. It involves transforming raw data into a clean and structured format suitable for analysis. The goal is to enhance data quality and ensure that the dataset accurately reflects the real-world phenomena being studied. In this project, preprocessing began with importing the dataset and identifying any missing or inconsistent values. Rows containing missing entries were removed to maintain the integrity of statistical results. Next, data types were verified and corrected where necessary to ensure that numerical operations could be performed on relevant columns like Survey_Stress_Score, Sleep_Hours, and Wearable_Stress_Score. Descriptive statistics were used to understand the spread and central tendencies of key variables. Finally, column names were standardized for ease of use. These steps ensure that the dataset is clean, reliable, and ready for insightful visualizations and analysis.

2.0.1 CREATE A DATA FRAME

[1]:
```
import pandas as pd
```

[3]:
```
df = pd.read_csv(r"C:/Users\athul\Downloads\mental_health_analysis.csv")
df
```

```
[3]:       User_ID  Age Gender  Social_Media_Hours  Exercise_Hours  Sleep_Hours  \
     0           1   16      F            9.654486        2.458001     5.198926
     1           2   17      M            9.158143        0.392095     8.866097
     2           3   15      M            5.028755        0.520119     4.943095
     3           4   17      F            7.951103        1.022630     5.262773
     4           5   17      F            1.357459        1.225462     6.196080
     …       …  …    …                 …             …          …
     4995     4996   14      M            0.088148        1.003339     8.684888
     4996     4997   15      F            7.161276        1.024644     5.312684
     4997     4998   14      M            3.444383        2.877972     9.227726
     4998     4999   18      F            7.866525        2.395839     4.317831
     4999     5000   18      M            3.389362        1.375646     8.693171

           Screen_Time_Hours  Survey_Stress_Score  Wearable_Stress_Score  \
     0              8.158189                    3               0.288962
     1              5.151993                    5               0.409446
     2              9.209325                    2               0.423837
     3              9.823658                    5               0.666021
     4             11.338990                    5               0.928060
     …                 …                  …                    …
     4995           5.922202                    1               0.750205
     4996          10.224924                    4               0.427209
     4997           4.059322                    4               0.002893
     4998          10.657076                    2               0.612063
     4999           6.977589                    5               0.952662

          Support_System Academic_Performance
     0          Moderate            Excellent
     1          Moderate                 Good
     2          Moderate                 Poor
     3          Moderate              Average
     4              High                 Poor
     …             …                  …
     4995       Moderate              Average
     4996       Moderate            Excellent
     4997           High                 Good
     4998           High              Average
     4999       Moderate            Excellent

     [5000 rows x 11 columns]
```

2.0.2 VIEW DATA

```
[5]: df.head()
```

```
[5]:    User_ID  Age Gender  Social_Media_Hours  Exercise_Hours  Sleep_Hours  \
     0        1   16      F            9.654486        2.458001     5.198926
```

```
1        2   17      M            9.158143        0.392095     8.866097
2        3   15      M            5.028755        0.520119     4.943095
3        4   17      F            7.951103        1.022630     5.262773
4        5   17      F            1.357459        1.225462     6.196080

   Screen_Time_Hours  Survey_Stress_Score  Wearable_Stress_Score  \
0           8.158189                    3               0.288962
1           5.151993                    5               0.409446
2           9.209325                    2               0.423837
3           9.823658                    5               0.666021
4          11.338990                    5               0.928060

  Support_System Academic_Performance
0       Moderate            Excellent
1       Moderate                 Good
2       Moderate                 Poor
3       Moderate              Average
4           High                 Poor
```

[7]:
```
df.tail()
```

[7]:
```
      User_ID  Age Gender  Social_Media_Hours  Exercise_Hours  Sleep_Hours  \
4995     4996   14      M            0.088148        1.003339     8.684888
4996     4997   15      F            7.161276        1.024644     5.312684
4997     4998   14      M            3.444383        2.877972     9.227726
4998     4999   18      F            7.866525        2.395839     4.317831
4999     5000   18      M            3.389362        1.375646     8.693171

      Screen_Time_Hours  Survey_Stress_Score  Wearable_Stress_Score  \
4995           5.922202                    1               0.750205
4996          10.224924                    4               0.427209
4997           4.059322                    4               0.002893
4998          10.657076                    2               0.612063
4999           6.977589                    5               0.952662

     Support_System Academic_Performance
4995       Moderate              Average
4996       Moderate            Excellent
4997           High                 Good
4998           High              Average
4999       Moderate            Excellent
```

[9]:
```
df.describe()
```

[9]:
```
           User_ID          Age  Social_Media_Hours  Exercise_Hours  \
count  5000.000000  5000.000000         5000.000000     5000.000000
mean   2500.500000    15.493200            4.932081        1.498151
```

```
std    1443.520003     1.715151            2.853928        0.873984
min       1.000000    13.000000            0.000528        0.000473
25%    1250.750000    14.000000            2.473150        0.734431
50%    2500.500000    16.000000            4.898176        1.483432
75%    3750.250000    17.000000            7.369195        2.276089
max    5000.000000    18.000000            9.995052        2.999774

       Sleep_Hours  Screen_Time_Hours  Survey_Stress_Score  \
count  5000.000000        5000.000000          5000.000000
mean      7.057370           7.068630             3.015800
std       1.722211           2.883494             1.414762
min       4.001515           2.000481             1.000000
25%       5.611836           4.574327             2.000000
50%       7.068874           7.118979             3.000000
75%       8.519411           9.526335             4.000000
max       9.999229          11.999010             5.000000

       Wearable_Stress_Score
count            5000.000000
mean                0.496618
std                 0.289768
min                 0.000102
25%                 0.244615
50%                 0.500404
75%                 0.749929
max                 0.999812
```

2.0.3 ACCESS COLUMNS

[13]:

```
df['Exercise_Hours']
```

[13]:

```
0       2.458001
1       0.392095
2       0.520119
3       1.022630
4       1.225462
          …
4995    1.003339
4996    1.024644
4997    2.877972
4998    2.395839
4999    1.375646
Name: Exercise_Hours, Length: 5000, dtype: float64
```

[15]:

```
df['Social_Media_Hours']
```

```
[15]: 0       9.654486
      1       9.158143
      2       5.028755
      3       7.951103
      4       1.357459
                ...
      4995    0.088148
      4996    7.161276
      4997    3.444383
      4998    7.866525
      4999    3.389362
      Name: Social_Media_Hours, Length: 5000, dtype: float64
```

2.0.4 ACCESS ROWS

```
[17]: df.iloc[0]
```

```
[17]: User_ID                          1
      Age                             16
      Gender                           F
      Social_Media_Hours        9.654486
      Exercise_Hours            2.458001
      Sleep_Hours               5.198926
      Screen_Time_Hours         8.158189
      Survey_Stress_Score              3
      Wearable_Stress_Score     0.288962
      Support_System            Moderate
      Academic_Performance     Excellent
      Name: 0, dtype: object
```

```
[19]: df.loc[1]
```

```
[19]: User_ID                         2
      Age                            17
      Gender                          M
      Social_Media_Hours       9.158143
      Exercise_Hours           0.392095
      Sleep_Hours              8.866097
      Screen_Time_Hours        5.151993
      Survey_Stress_Score             5
      Wearable_Stress_Score    0.409446
      Support_System           Moderate
      Academic_Performance         Good
      Name: 1, dtype: object
```

2.0.5 MANIPULATTE DATA

```
[23]: df['Sleep_Hours'] = df['Survey_Stress_Score'] / df['Age']
      df.head()
```

```
[23]:    User_ID  Age Gender  Social_Media_Hours  Exercise_Hours  Sleep_Hours  \
      0        1   16      F            9.654486        2.458001     0.187500
      1        2   17      M            9.158143        0.392095     0.294118
      2        3   15      M            5.028755        0.520119     0.133333
      3        4   17      F            7.951103        1.022630     0.294118
      4        5   17      F            1.357459        1.225462     0.294118

         Screen_Time_Hours  Survey_Stress_Score  Wearable_Stress_Score  \
      0           8.158189                    3               0.288962
      1           5.151993                    5               0.409446
      2           9.209325                    2               0.423837
      3           9.823658                    5               0.666021
      4          11.338990                    5               0.928060

        Support_System Academic_Performance
      0       Moderate            Excellent
      1       Moderate                 Good
      2       Moderate                 Poor
      3       Moderate              Average
      4           High                 Poor
```

2.0.6 inplace=True vs inplace=False in Pandas

2.0.7 Understanding the inplace Parameter in Pandas Functions

When using pandas functions like .dropna(), .fillna(), .drop(), etc., the inplace parameter decides whether to modify the DataFrame directly or return a new one.

inplace=True (Modifies the Original DataFrame)

- Changes are made directly to the existing DataFrame.
- No need to assign the result to a new variable.

inplace=False (Creates a New DataFrame)

- The function returns a new DataFrame with changes.
- You must assign it to a variable if you want to keep the changes.

```
[25]: df.dropna(inplace=True)
      print("After dropna (inplace=True):")
      df.head()
```

```
After dropna (inplace=True):
```

[25]:
```
   User_ID  Age Gender  Social_Media_Hours  Exercise_Hours  Sleep_Hours  \
0        1   16      F            9.654486        2.458001     0.187500
1        2   17      M            9.158143        0.392095     0.294118
2        3   15      M            5.028755        0.520119     0.133333
3        4   17      F            7.951103        1.022630     0.294118
4        5   17      F            1.357459        1.225462     0.294118

   Screen_Time_Hours  Survey_Stress_Score  Wearable_Stress_Score  \
0           8.158189                    3               0.288962
1           5.151993                    5               0.409446
2           9.209325                    2               0.423837
3           9.823658                    5               0.666021
4          11.338990                    5               0.928060

  Support_System Academic_Performance
0       Moderate            Excellent
1       Moderate                 Good
2       Moderate                 Poor
3       Moderate              Average
4           High                 Poor
```

[27]:
```python
df.dropna(inplace=False)
print("After dropna (inplace=False):")
df.head()
```

```
After dropna (inplace=False):
```

[27]:
```
   User_ID  Age Gender  Social_Media_Hours  Exercise_Hours  Sleep_Hours  \
0        1   16      F            9.654486        2.458001     0.187500
1        2   17      M            9.158143        0.392095     0.294118
2        3   15      M            5.028755        0.520119     0.133333
3        4   17      F            7.951103        1.022630     0.294118
4        5   17      F            1.357459        1.225462     0.294118

   Screen_Time_Hours  Survey_Stress_Score  Wearable_Stress_Score  \
0           8.158189                    3               0.288962
1           5.151993                    5               0.409446
2           9.209325                    2               0.423837
3           9.823658                    5               0.666021
4          11.338990                    5               0.928060

  Support_System Academic_Performance
0       Moderate            Excellent
1       Moderate                 Good
2       Moderate                 Poor
3       Moderate              Average
4           High                 Poor
```

2.0.8 HOW TO SET INDEX IN A DATA FRAME

[31]:
```
df.set_index('Gender', inplace=True)
df.head()
```

[31]:
```
        User_ID  Age  Social_Media_Hours  Exercise_Hours  Sleep_Hours  \
Gender
F             1   16            9.654486        2.458001     0.187500
M             2   17            9.158143        0.392095     0.294118
M             3   15            5.028755        0.520119     0.133333
F             4   17            7.951103        1.022630     0.294118
F             5   17            1.357459        1.225462     0.294118

        Screen_Time_Hours  Survey_Stress_Score  Wearable_Stress_Score  \
Gender
F                8.158189                    3               0.288962
M                5.151993                    5               0.409446
M                9.209325                    2               0.423837
F                9.823658                    5               0.666021
F               11.338990                    5               0.928060

       Support_System Academic_Performance
Gender
F            Moderate            Excellent
M            Moderate                 Good
M            Moderate                 Poor
F            Moderate              Average
F                High                 Poor
```

[33]:
```
df.reset_index(inplace=True)
print("DataFrame after resetting the index:")
df.head()
```

```
DataFrame after resetting the index:
```

[33]:
```
  Gender  User_ID  Age  Social_Media_Hours  Exercise_Hours  Sleep_Hours  \
0      F        1   16            9.654486        2.458001     0.187500
1      M        2   17            9.158143        0.392095     0.294118
2      M        3   15            5.028755        0.520119     0.133333
3      F        4   17            7.951103        1.022630     0.294118
4      F        5   17            1.357459        1.225462     0.294118

   Screen_Time_Hours  Survey_Stress_Score  Wearable_Stress_Score  \
0           8.158189                    3               0.288962
1           5.151993                    5               0.409446
2           9.209325                    2               0.423837
3           9.823658                    5               0.666021
4          11.338990                    5               0.928060
```

```
  Support_System Academic_Performance
0       Moderate            Excellent
1       Moderate                 Good
2       Moderate                 Poor
3       Moderate              Average
4           High                 Poor
```

[35]:

```
print(df.loc[:, 'Support_System'])
```

```
0       Moderate
1       Moderate
2       Moderate
3       Moderate
4           High
          …
4995    Moderate
4996    Moderate
4997        High
4998        High
4999    Moderate
Name: Support_System, Length: 5000, dtype: object
```

[37]:

```
print(df.iloc[:, 2])
```

```
0       16
1       17
2       15
3       17
4       17
        ..
4995    14
4996    15
4997    14
4998    18
4999    18
Name: Age, Length: 5000, dtype: int64
```

[39]:

```
df_cleaned = df.dropna()
print(df_cleaned)
```

```
     Gender  User_ID  Age  Social_Media_Hours  Exercise_Hours  Sleep_Hours  \
0         F        1   16            9.654486        2.458001     0.187500
1         M        2   17            9.158143        0.392095     0.294118
2         M        3   15            5.028755        0.520119     0.133333
3         F        4   17            7.951103        1.022630     0.294118
4         F        5   17            1.357459        1.225462     0.294118
…     …      …  …                 …             …          …
```

```
4995      M     4996    14          0.088148         1.003339      0.071429
4996      F     4997    15          7.161276         1.024644      0.266667
4997      M     4998    14          3.444383         2.877972      0.285714
4998      F     4999    18          7.866525         2.395839      0.111111
4999      M     5000    18          3.389362         1.375646      0.277778

      Screen_Time_Hours  Survey_Stress_Score  Wearable_Stress_Score  \
0              8.158189                    3               0.288962
1              5.151993                    5               0.409446
2              9.209325                    2               0.423837
3              9.823658                    5               0.666021
4             11.338990                    5               0.928060
…                   …                  …                    …
4995           5.922202                    1               0.750205
4996          10.224924                    4               0.427209
4997           4.059322                    4               0.002893
4998          10.657076                    2               0.612063
4999           6.977589                    5               0.952662

     Support_System Academic_Performance
0          Moderate            Excellent
1          Moderate                 Good
2          Moderate                 Poor
3          Moderate              Average
4              High                 Poor
…              …                 …
4995       Moderate              Average
4996       Moderate            Excellent
4997           High                 Good
4998           High              Average
4999       Moderate            Excellent

[5000 rows x 11 columns]
```

[41]:

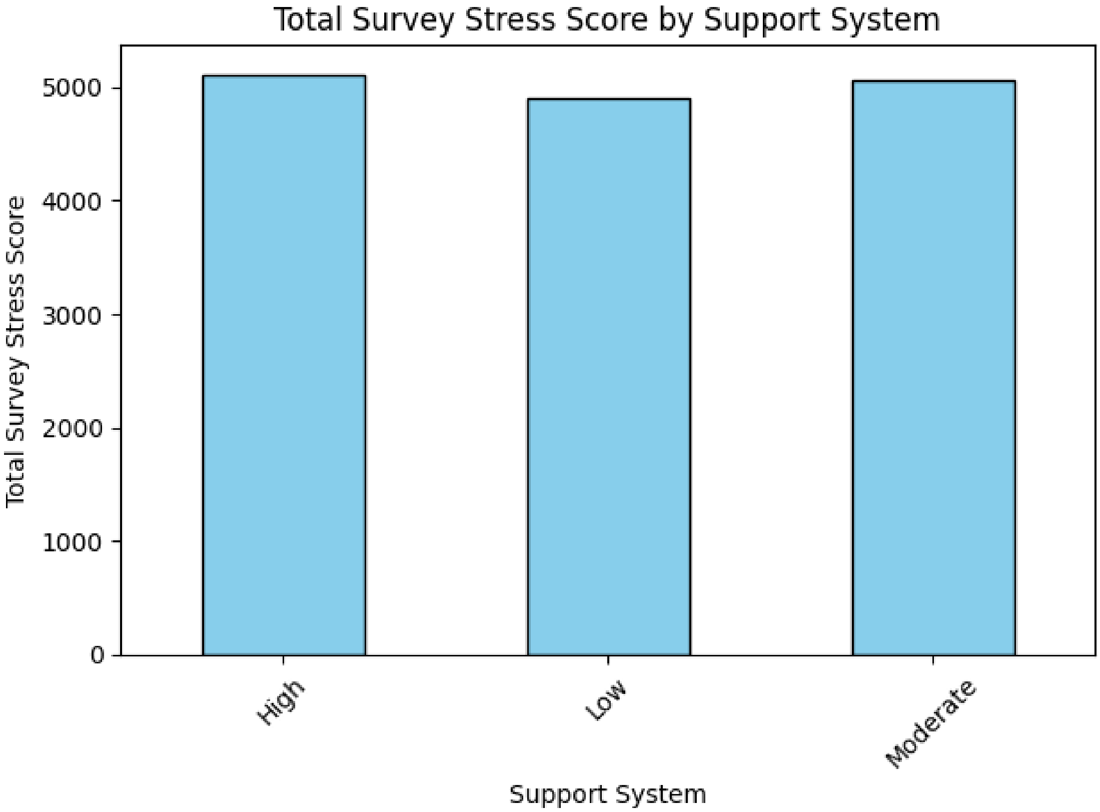
```
print("Missing values in each column before cleaning:")
print(df.isnull().sum())
```

```
Missing values in each column before cleaning:
Gender                   0
User_ID                  0
Age                      0
Social_Media_Hours       0
Exercise_Hours           0
Sleep_Hours              0
Screen_Time_Hours        0
Survey_Stress_Score      0
Wearable_Stress_Score    0
Support_System           0
```

```
Academic_Performance     0
dtype: int64
```

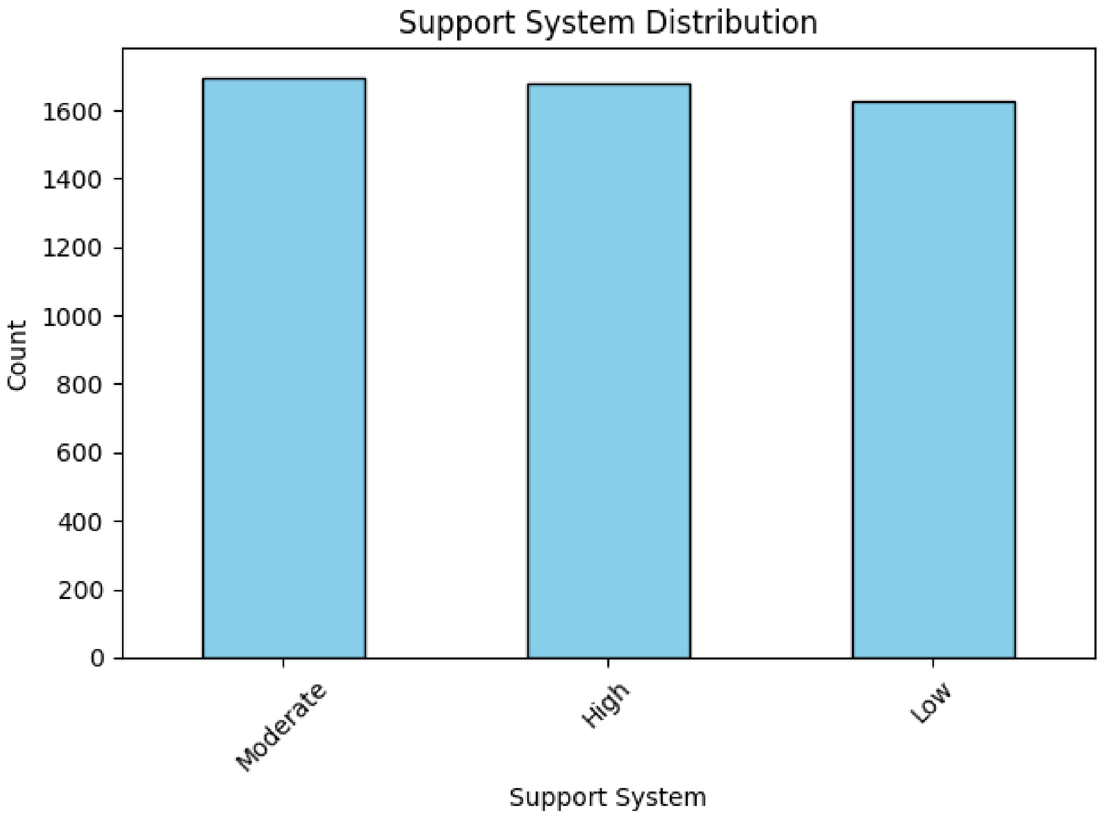
[58]:

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

# Load the dataset
df = pd.read_csv(r"C:\Users\athul\Downloads\mental_health_analysis.csv")

# Create a summary: total Survey Stress Score by Support System
stress_summary = df.groupby("Support_System")["Survey_Stress_Score"].sum()

# Plotting
stress_summary.plot(kind="bar", color="skyblue", edgecolor="black")
plt.title("Total Survey Stress Score by Support System")
plt.xlabel("Support System")
plt.ylabel("Total Survey Stress Score")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[60]:

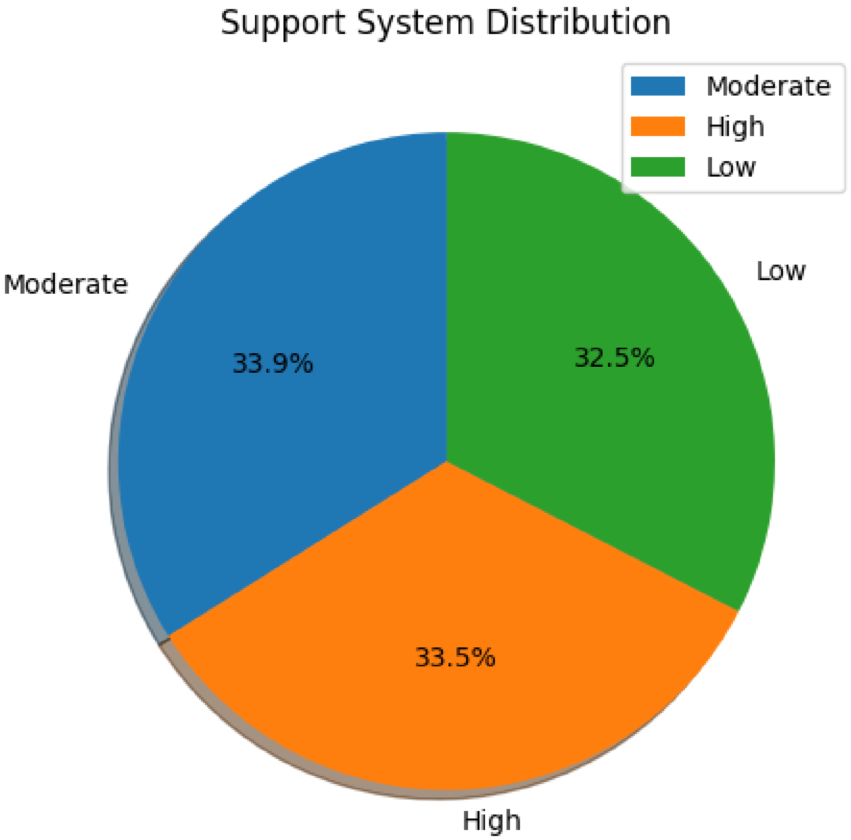
```
support_system_counts = df['Support_System'].value_counts()
support_system_counts.plot(kind="bar", color="skyblue", edgecolor="black")
plt.title("Support System Distribution")
plt.xlabel("Support System")
plt.ylabel("Count")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



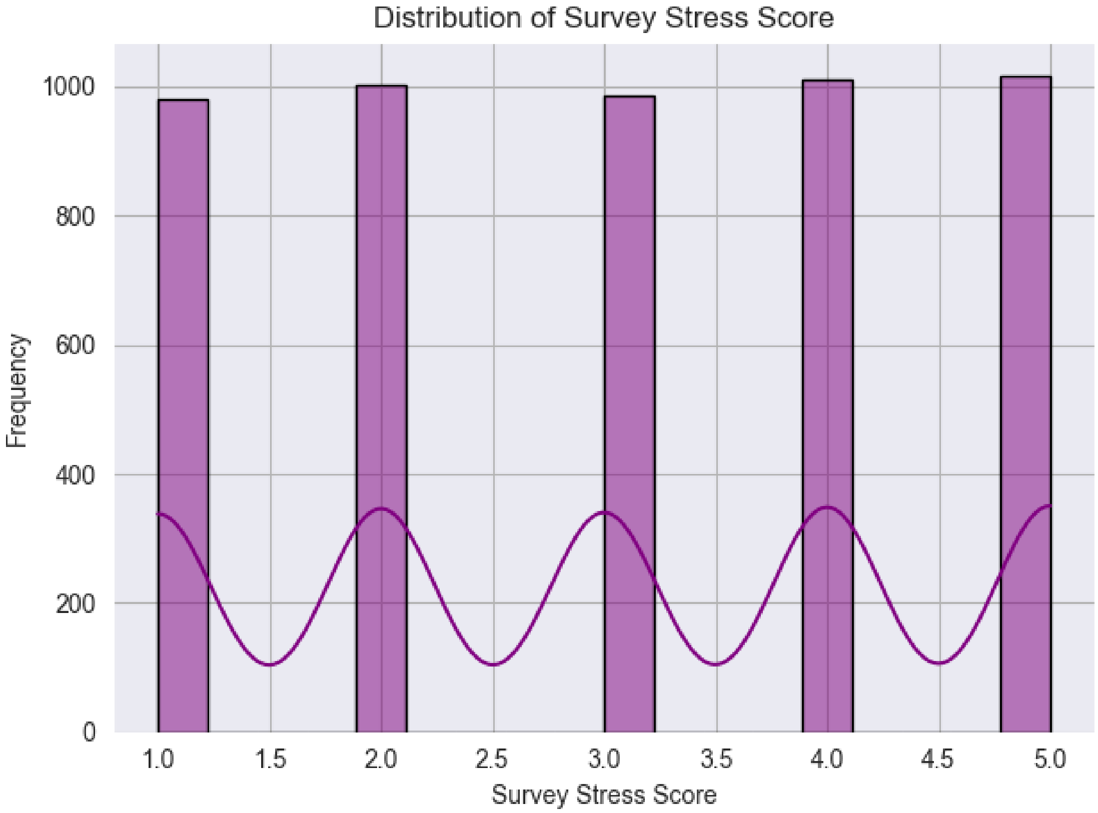
[62]:

```
# Count the occurrences of each Support System type
support_system_counts = df['Support_System'].value_counts()

# Plotting a pie chart
support_system_counts.plot(
    kind='pie',
    autopct='%1.1f%%',
    legend=True,
    title='Support System Distribution',
    shadow=True,
    startangle=90
)
```

```
plt.ylabel("")  # Hide y-axis label
plt.tight_layout()
plt.show()
```

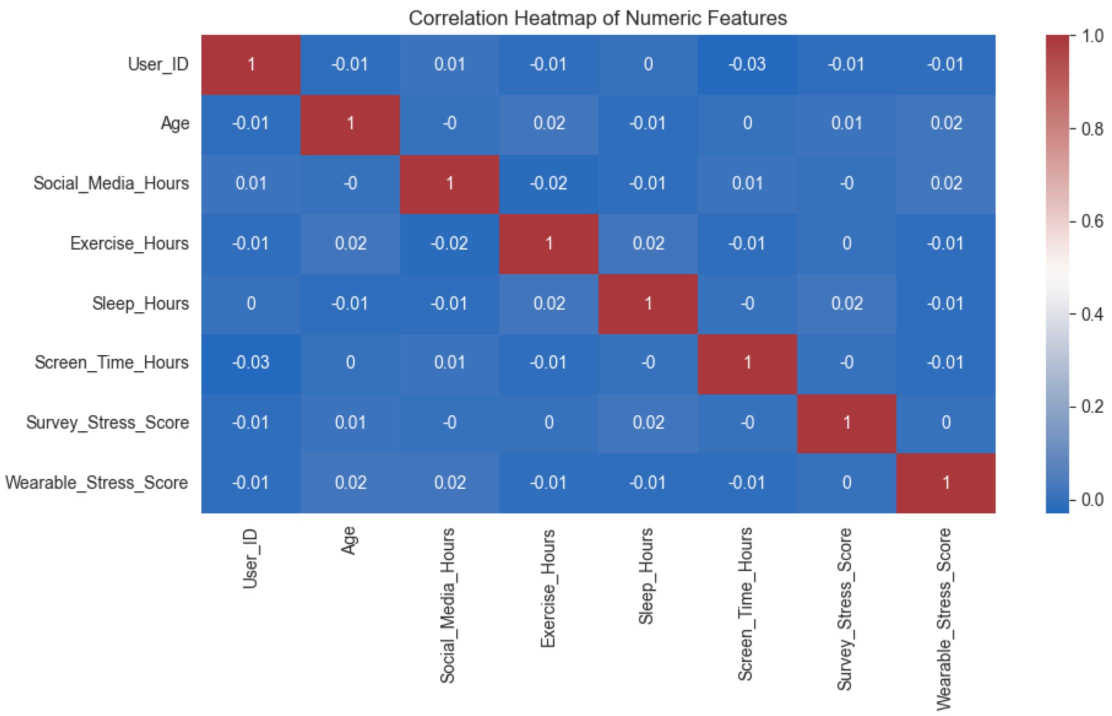


[66]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')

# Style settings
sns.set_style('darkgrid', {"grid.color": "0.7", "grid.linestyle": "-"})

# Plotting the distribution of Survey Stress Score
sns.histplot(df['Survey_Stress_Score'], kde=True, color='purple',␣
 ↪edgecolor='black')
plt.xlabel("Survey Stress Score")
plt.ylabel("Frequency")
plt.title("Distribution of Survey Stress Score")
plt.tight_layout()
plt.show()
```

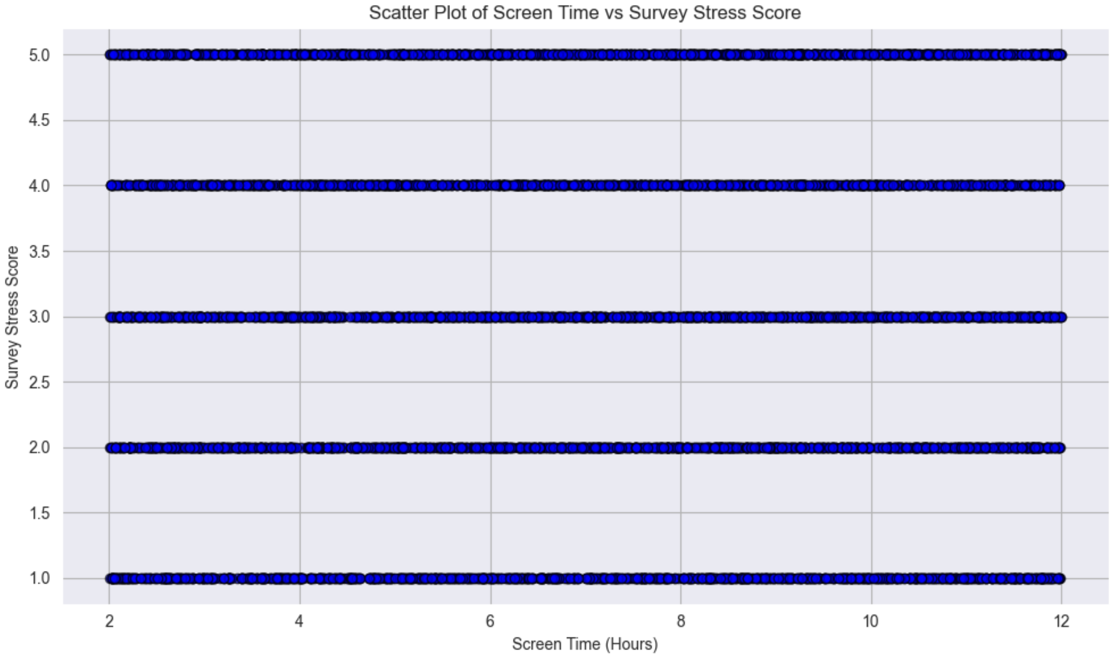


[68]:

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

# Plot the heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(
    correlation_matrix.round(2),
    annot=True,
    cmap=sns.color_palette("vlag", as_cmap=True)
)
plt.title("Correlation Heatmap of Numeric Features")
plt.tight_layout()
plt.show()
```



[70]:
```
df.dropna(inplace=True)

# Scatter plot: Screen Time vs Survey Stress Score
plt.figure(figsize=(10, 6))
plt.scatter(df['Screen_Time_Hours'], df['Survey_Stress_Score'], alpha=0.7,␣
  ↪c='blue', edgecolor='k')
plt.xlabel("Screen Time (Hours)")
plt.ylabel("Survey Stress Score")
plt.title("Scatter Plot of Screen Time vs Survey Stress Score")
plt.grid(True)
plt.tight_layout()
plt.show()
```



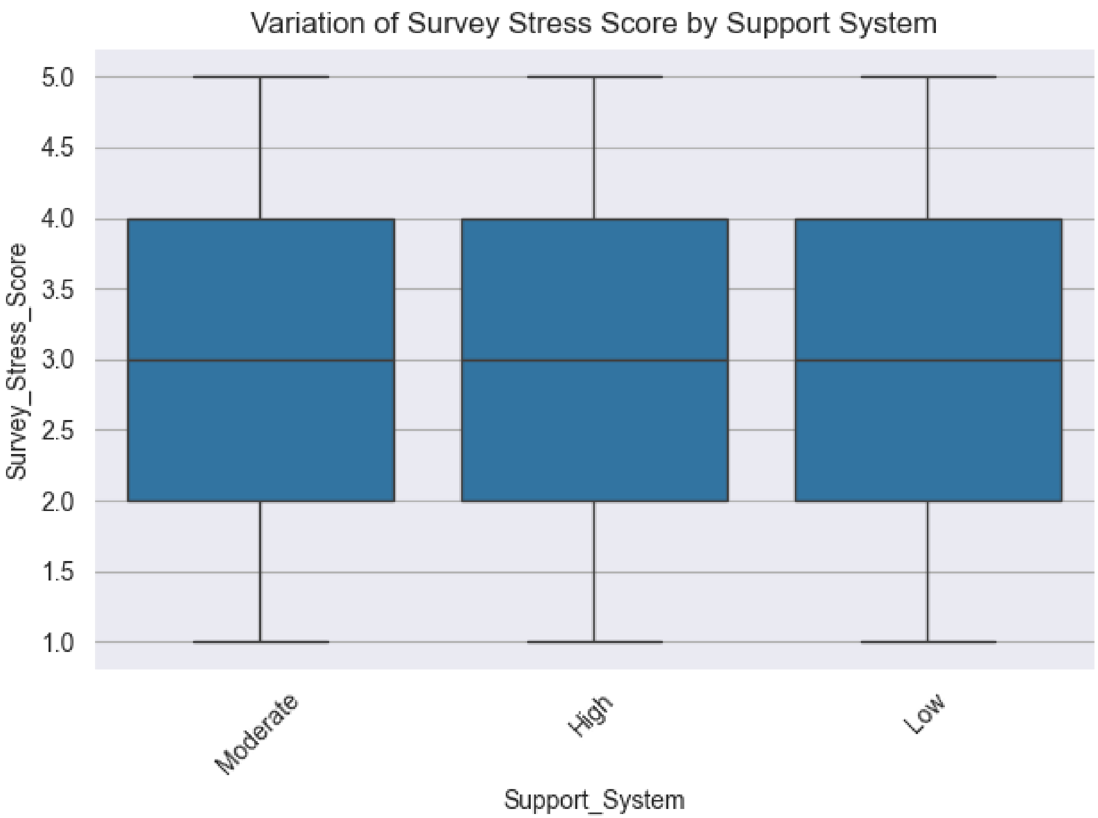
```
[72]: df.isna()
```

```
[72]:       User_ID    Age  Gender  Social_Media_Hours  Exercise_Hours  Sleep_Hours  \
      0       False  False   False               False           False        False
      1       False  False   False               False           False        False
      2       False  False   False               False           False        False
      3       False  False   False               False           False        False
      4       False  False   False               False           False        False
      …       …    …     …                  …             …          …
      4995    False  False   False               False           False        False
      4996    False  False   False               False           False        False
      4997    False  False   False               False           False        False
      4998    False  False   False               False           False        False
      4999    False  False   False               False           False        False

            Screen_Time_Hours  Survey_Stress_Score  Wearable_Stress_Score  \
      0                 False                False                  False
      1                 False                False                  False
      2                 False                False                  False
      3                 False                False                  False
      4                 False                False                  False
      …                  …                  …                    …
      4995              False                False                  False
      4996              False                False                  False
      4997              False                False                  False
```

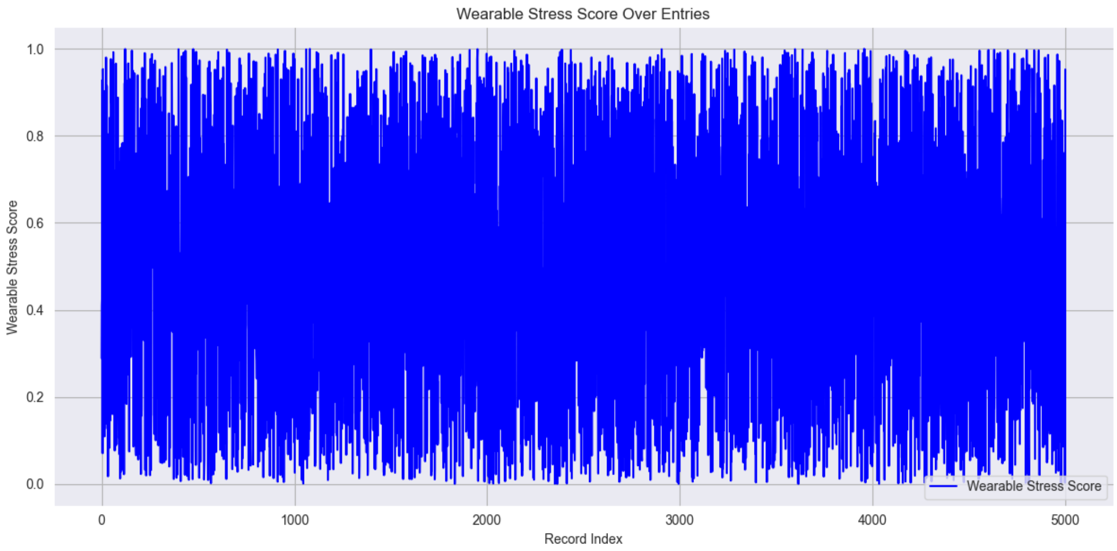
```
4998                  False                False                  False
4999                  False                False                  False

      Support_System  Academic_Performance
0               False                 False
1               False                 False
2               False                 False
3               False                 False
4               False                 False
…               …                   …
4995            False                 False
4996            False                 False
4997            False                 False
4998            False                 False
4999            False                 False

[5000 rows x 11 columns]
```

[74]:

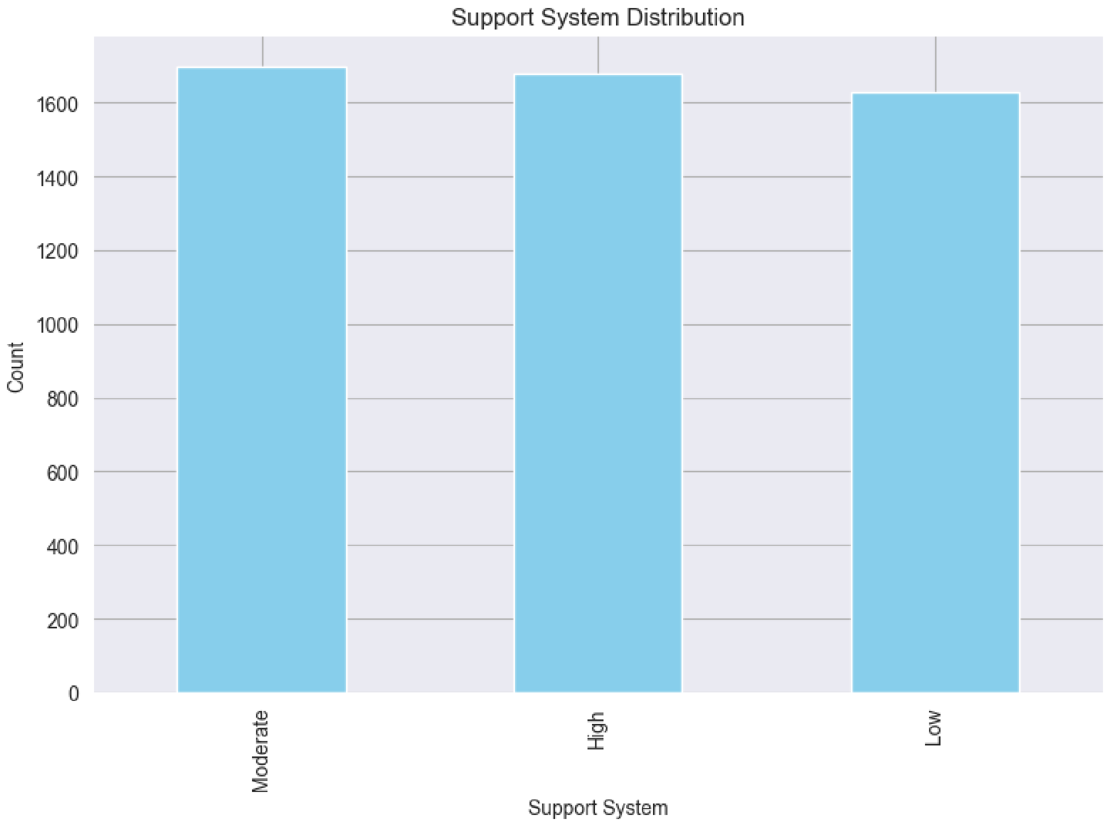
```python
df.dropna(subset=["Support_System", "Survey_Stress_Score"], inplace=True)

# Create the boxplot
sns.boxplot(x="Support_System", y="Survey_Stress_Score", data=df)
plt.title("Variation of Survey Stress Score by Support System")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



[76]:
```
df.dropna(subset=["Wearable_Stress_Score"], inplace=True)

# Plotting Wearable Stress Score over index
plt.figure(figsize=(12, 6))
plt.plot(df.index, df["Wearable_Stress_Score"], label="Wearable Stress Score",␣
 ↪color="blue")
plt.xlabel("Record Index")
plt.ylabel("Wearable Stress Score")
plt.title("Wearable Stress Score Over Entries")
plt.legend()
plt.tight_layout()
plt.show()
```

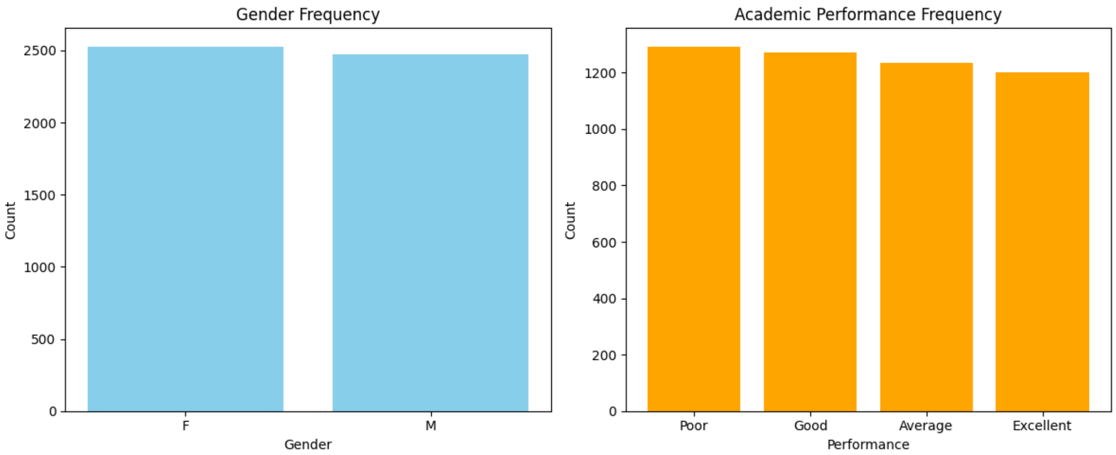


[78]:
```
df.dropna(subset=["Support_System"], inplace=True)

# Plot the distribution of Support System
df["Support_System"].value_counts().plot(kind="bar", color="skyblue",␣
 ↪figsize=(8, 6))
plt.title("Support System Distribution")
plt.xlabel("Support System")
plt.ylabel("Count")
plt.tight_layout()
plt.show()
```



[7]:
```
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_csv(r"C:\Users\athul\Downloads\mental_health_analysis.csv")  #␣
↪Make sure this CSV is in your working directory

# Count frequency of Gender and Academic Performance
gender_counts = df['Gender'].value_counts()
performance_counts = df['Academic_Performance'].value_counts()

plt.figure(figsize=(12, 5))

# Gender frequency
plt.subplot(1, 2, 1)
plt.bar(gender_counts.index, gender_counts.values, color='skyblue')
plt.title("Gender Frequency")
plt.xlabel("Gender")
plt.ylabel("Count")

# Academic Performance frequency
```

```
plt.subplot(1, 2, 2)
plt.bar(performance_counts.index, performance_counts.values, color='orange')
plt.title("Academic Performance Frequency")
plt.xlabel("Performance")
plt.ylabel("Count")

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



[9]:

```
# Step 2: Min-Max Normalization Function
def min_max_normalization(column):
    X_min = min(column)
    X_max = max(column)
    return [(x - X_min) / (X_max - X_min) for x in column]

# Step 3: Select numeric columns and create normalized DataFrame
numerical_columns = df.select_dtypes(include=['number']).columns
normalized_df = df.copy()

# Step 4: Normalize and show original + normalized values
for col in numerical_columns:
    data = df[col].tolist()
    normalized_data = min_max_normalization(data)
    normalized_df[col] = normalized_data
    print(f"Original Data for '{col}':", data[:5])
    print(f"Min-Max Normalized Data for '{col}':", normalized_data[:5])
    print()

# Step 5: Display first 5 rows of normalized data
print("Normalized DataFrame (first 5 rows):")
print(normalized_df[numerical_columns].head())
```

```
Original Data for 'User_ID': [1, 2, 3, 4, 5]
Min-Max Normalized Data for 'User_ID': [0.0, 0.00020004000800160032,
0.00040008001600320064, 0.000600120024004801, 0.0008001600320064013]

Original Data for 'Age': [16, 17, 15, 17, 17]
Min-Max Normalized Data for 'Age': [0.6, 0.8, 0.4, 0.8, 0.8]

Original Data for 'Social_Media_Hours': [9.654486346, 9.158143482, 5.028755201,
7.951102825, 1.357458531]
Min-Max Normalized Data for 'Social_Media_Hours': [0.9659247840441446,
0.9162633011687148, 0.5030982084942248, 0.7954930972123768, 0.1357673774664471]

Original Data for 'Exercise_Hours': [2.458001257, 0.392094761, 0.52011947,
1.022629619, 1.225462167]
Min-Max Normalized Data for 'Exercise_Hours': [0.8193668529553232,
0.13057092131785153, 0.17325576638659526, 0.3407981730288655,
0.4084247738148008]

Original Data for 'Sleep_Hours': [5.198925522, 8.866096662, 4.94309483,
5.262773303, 6.196080351]
Min-Max Normalized Data for 'Sleep_Hours': [0.19964453362823228,
0.811072687639666, 0.15698983283097753, 0.21028988651311475,
0.36590035128256104]

Original Data for 'Screen_Time_Hours': [8.158188998, 5.151993467, 9.209325483,
9.823657952, 11.33898971]
Min-Max Normalized Data for 'Screen_Time_Hours': [0.6158613979955446,
0.3151976031501909, 0.7209905159193469, 0.7824328038615463, 0.9339882805801236]

Original Data for 'Survey_Stress_Score': [3, 5, 2, 5, 5]
Min-Max Normalized Data for 'Survey_Stress_Score': [0.5, 1.0, 0.25, 1.0, 1.0]

Original Data for 'Wearable_Stress_Score': [0.288962247, 0.409446165,
0.423837485, 0.666020828, 0.928060356]
Min-Max Normalized Data for 'Wearable_Stress_Score': [0.28894366332821025,
0.40946252999175964, 0.4238580244692257, 0.6661116173773651, 0.9282271549515286]

Normalized DataFrame (first 5 rows):
   User_ID  Age  Social_Media_Hours  Exercise_Hours  Sleep_Hours  \
0   0.0000  0.6            0.965925        0.819367     0.199645
1   0.0002  0.8            0.916263        0.130571     0.811073
2   0.0004  0.4            0.503098        0.173256     0.156990
3   0.0006  0.8            0.795493        0.340798     0.210290
4   0.0008  0.8            0.135767        0.408425     0.365900

   Screen_Time_Hours  Survey_Stress_Score  Wearable_Stress_Score
0           0.615861                 0.50               0.288944
1           0.315198                 1.00               0.409463
```

```
2           0.720991                 0.25               0.423858
3           0.782433                 1.00               0.666112
4           0.933988                 1.00               0.928227
```

[11]:

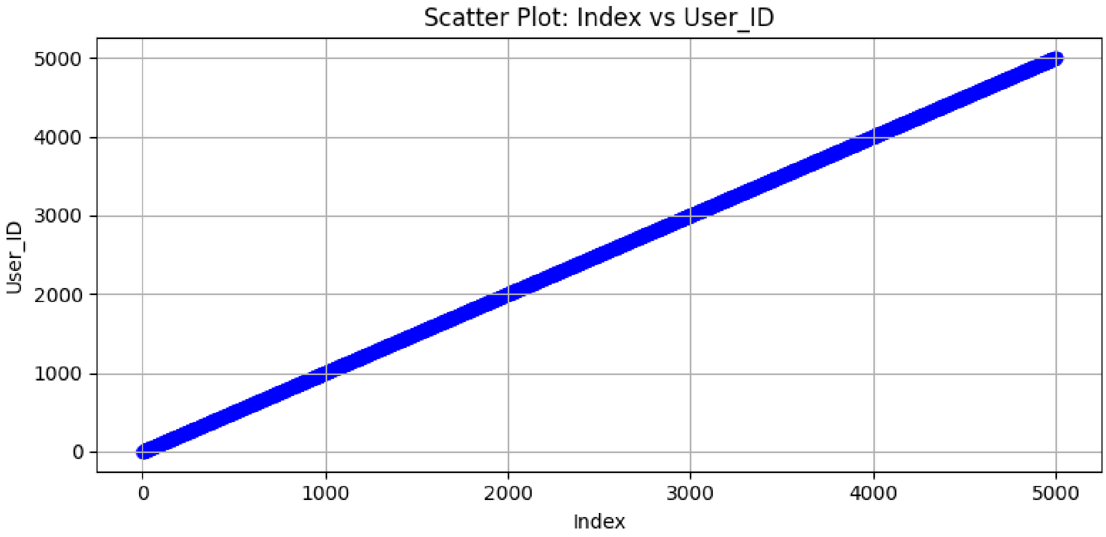
```python
import pandas as pd
from sklearn.preprocessing import StandardScaler

# Load the dataset
df = pd.read_csv(r"C:\Users\athul\Downloads\mental_health_analysis.csv")

# Select only numerical columns
numerical_columns = df.select_dtypes(include=['number']).columns
numerical_data = df[numerical_columns]

# Apply StandardScaler for z-score normalization
scaler = StandardScaler()
standardized_data = scaler.fit_transform(numerical_data)

# Create a new DataFrame with standardized values
standardized_df = pd.DataFrame(standardized_data, columns=numerical_columns)

# Display the original and standardized data
print("Original Data (first 5 rows):")
print(numerical_data.head())

print("\nStandardized Data (first 5 rows):")
print(standardized_df.head())
```

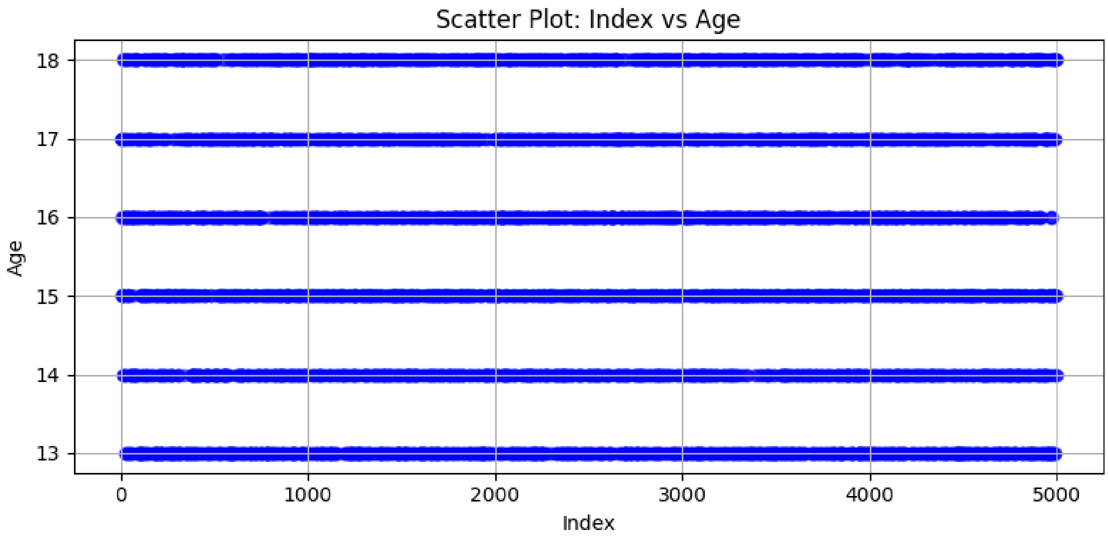
```
Original Data (first 5 rows):
   User_ID  Age  Social_Media_Hours  Exercise_Hours  Sleep_Hours  \
0        1   16            9.654486        2.458001     5.198926
1        2   17            9.158143        0.392095     8.866097
2        3   15            5.028755        0.520119     4.943095
3        4   17            7.951103        1.022630     5.262773
4        5   17            1.357459        1.225462     6.196080

   Screen_Time_Hours  Survey_Stress_Score  Wearable_Stress_Score
0           8.158189                    3               0.288962
1           5.151993                    5               0.409446
2           9.209325                    2               0.423837
3           9.823658                    5               0.666021
4          11.338990                    5               0.928060

Standardized Data (first 5 rows):
    User_ID       Age  Social_Media_Hours  Exercise_Hours  Sleep_Hours  \
0 -1.731704  0.295514            1.654869        1.098356    -1.079212
1 -1.731012  0.878611            1.480936       -1.265659     1.050340
2 -1.730319 -0.287584            0.033878       -1.119161    -1.227774
```

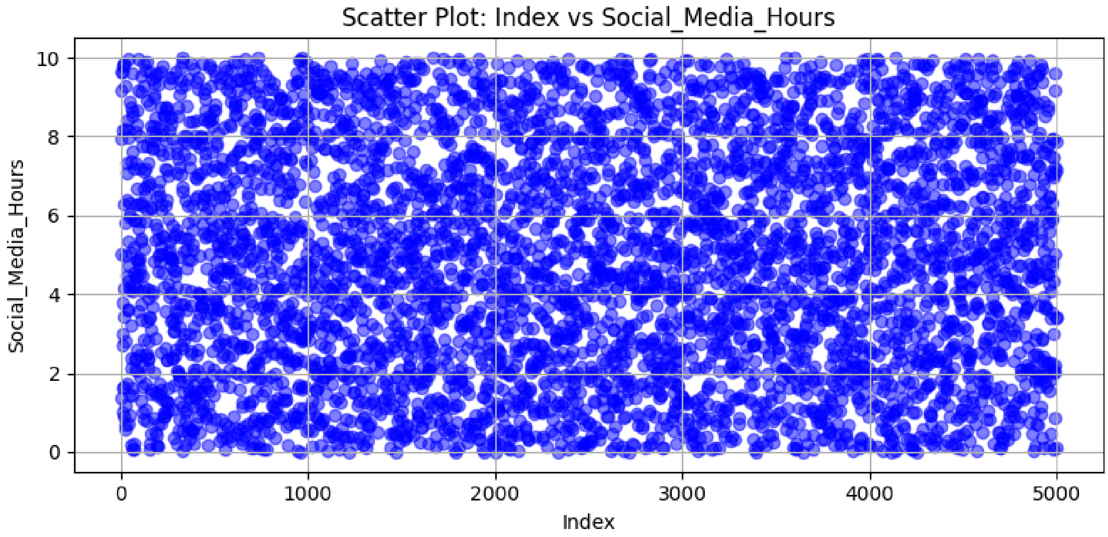
```
3 -1.729626  0.878611            1.057954       -0.544139     -1.042135
4 -1.728933  0.878611           -1.252652       -0.312038     -0.500157

   Screen_Time_Hours  Survey_Stress_Score  Wearable_Stress_Score
0           0.377898            -0.011169              -0.716698
1          -0.664759             1.402638              -0.300863
2           0.742471            -0.718073              -0.251193
3           0.955543             1.402638               0.584673
4           1.481115             1.402638               1.489070
```

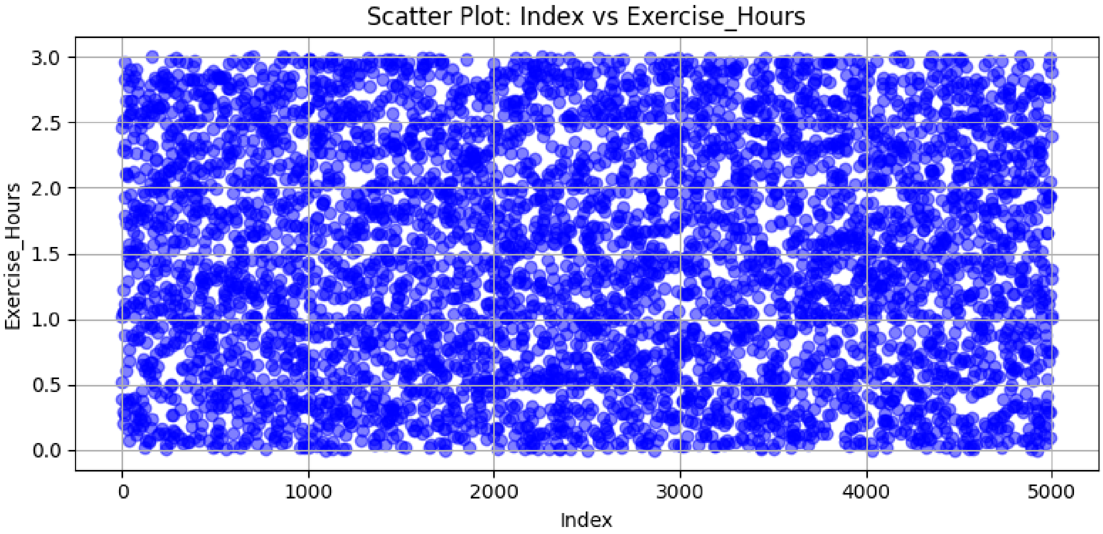
[25]:

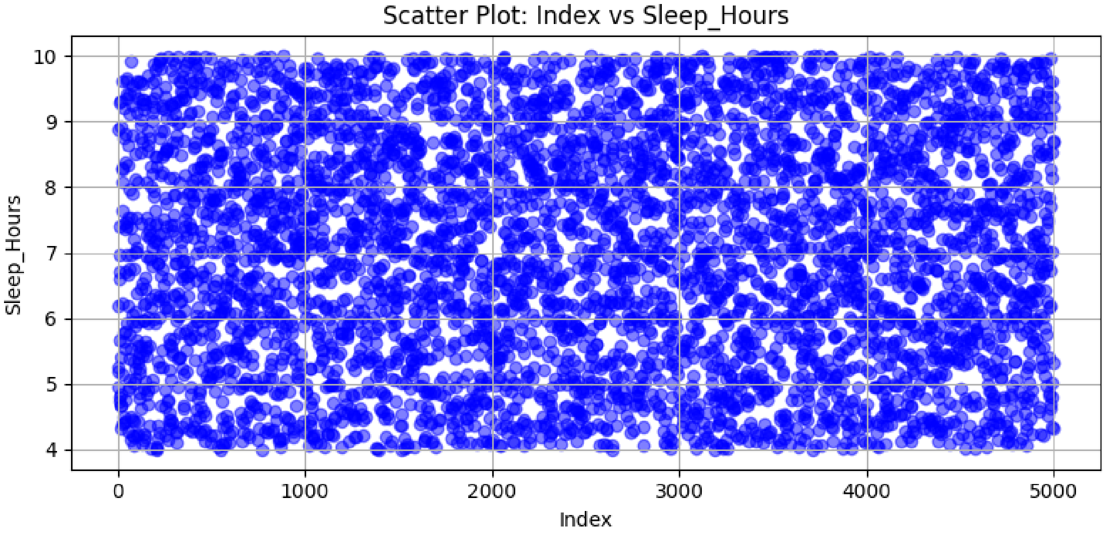
```
numerical_columns = df.select_dtypes(include=['number']).columns

# Create scatter plots for each numerical column
for col in numerical_columns:
    plt.figure(figsize=(8, 4))
    plt.scatter(df.index, df[col], color='blue', alpha=0.5)
    plt.title(f'Scatter Plot: Index vs {col}')
    plt.xlabel('Index')
    plt.ylabel(col)
    plt.grid(True)
    plt.tight_layout()
    plt.show()
```

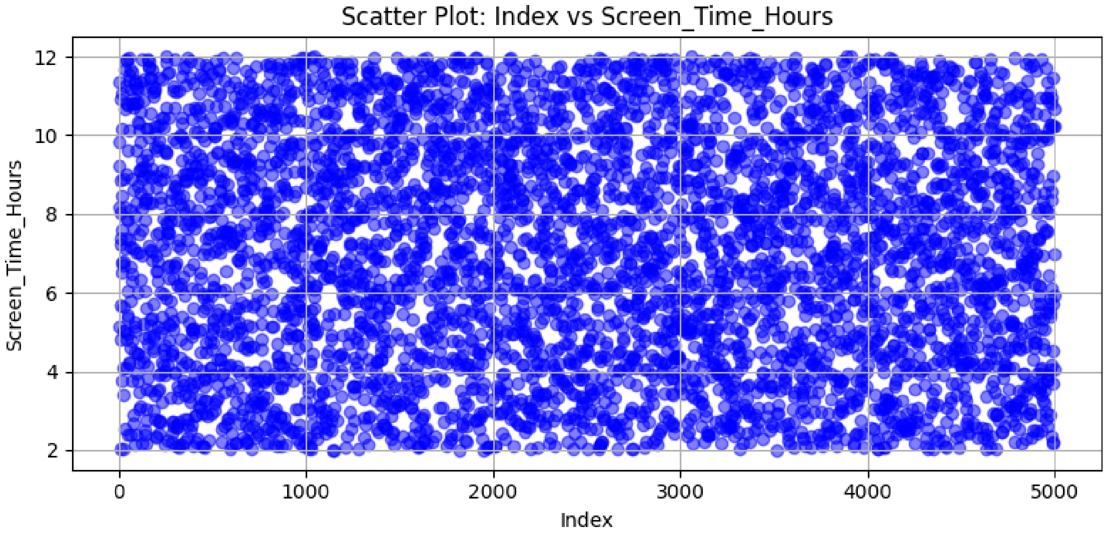


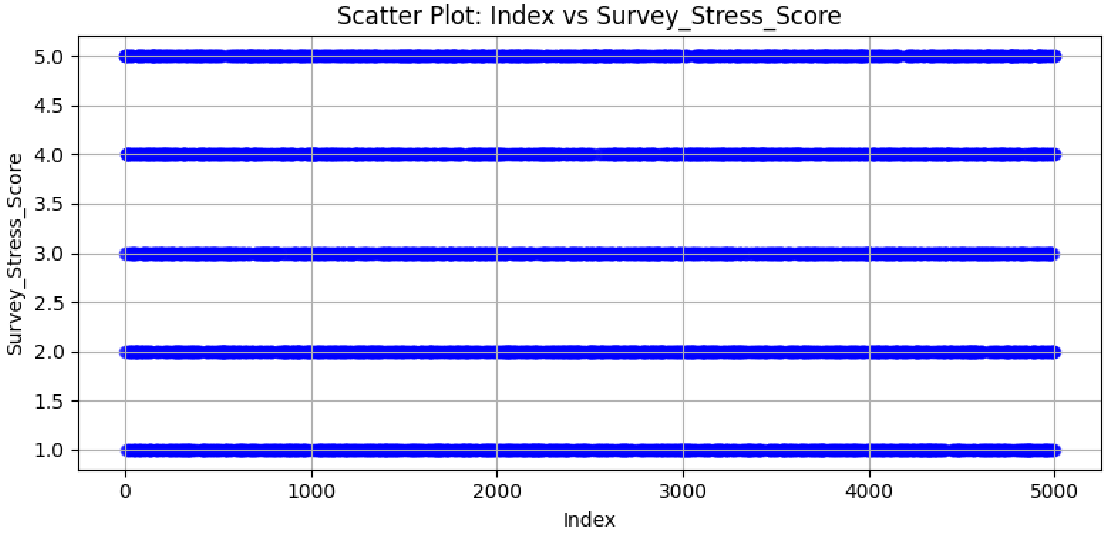

Scatter Plot: Index vs Age
Age
18
17
16
15
14
13
0
1000
2000
3000
4000
5000
Index

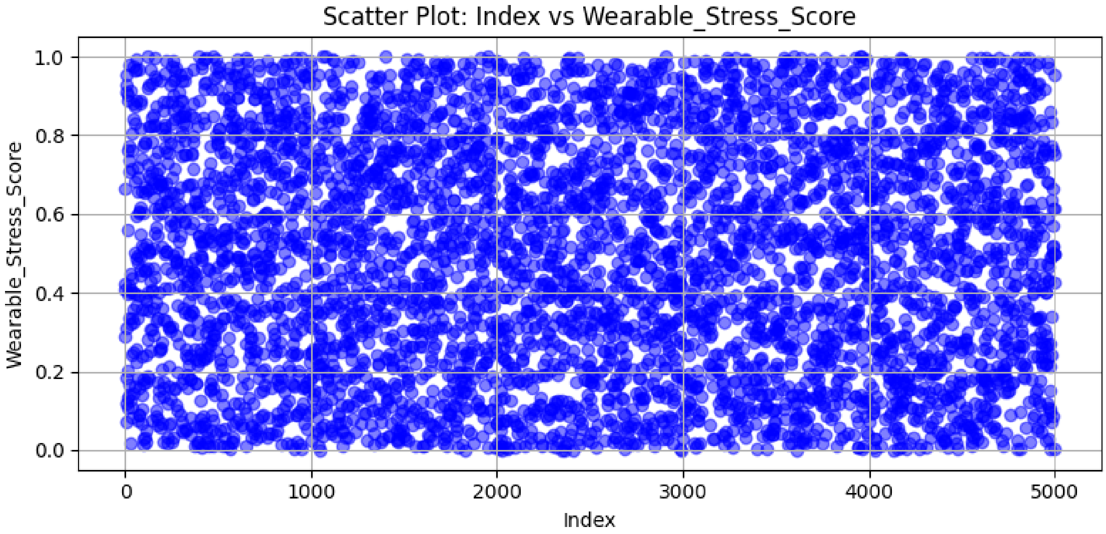

Scatter Plot: Index vs Social_Media_Hours
Social_Media_Hours
10
8
6
4
2
0
0
1000
2000
3000
4000
5000
Index


Scatter Plot: Index vs Exercise_Hours
Exercise_Hours
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
1000
2000
3000
4000
5000
Index


Scatter Plot: Index vs Sleep_Hours
Sleep_Hours
10
9
8
7
6
5
4
0
1000
2000
3000
4000
5000
Index


Scatter Plot: Index vs Screen_Time_Hours
Screen_Time_Hours
2
4
6
8
10
12
0
1000
2000
3000
4000
5000
Index


Scatter Plot: Index vs Survey_Stress_Score
Survey_Stress_Score
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
0
1000
2000
3000
4000
5000
Index


Scatter Plot: Index vs Wearable_Stress_Score
Wearable_Stress_Score
0.0
0.2
0.4
0.6
0.8
1.0
0
1000
2000
3000
4000
5000
Index