```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

```
df=pd.read_csv("/content/Sleep_Data.csv")
```

```
df.columns
```

```
     Index(['Gender', 'Trouble_Sleep', 'Refreshing_sleep', 'Age ',
            'Trouble_Stay_awake', 'No_of_hours_in_Sleeping'],
           dtype='object')
```

```
df = df[(df['Age '] >= 3) & (df['Age '] <= 8)]

df = df[(df['Trouble_Sleep'] >= 1) & (df['Trouble_Sleep'] <6)]
df = df[(df['Refreshing_sleep'] >= 1) & (df['Refreshing_sleep'] < 6)]
df = df[(df['Trouble_Stay_awake'] >= 1) & (df['Trouble_Stay_awake'] < 6)]
df = df[(df['No_of_hours_in_Sleeping'] >= 1) & (df['No_of_hours_in_Sleeping'] <=10)]
```

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

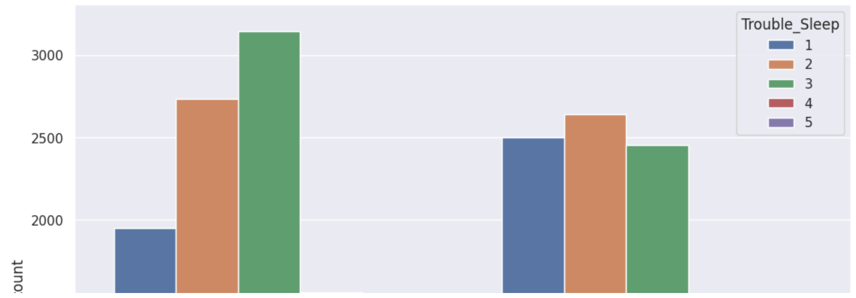
```
     Gender                     0
     Trouble_Sleep              0
     Refreshing_sleep           0
     Age                        0
     Trouble_Stay_awake         0
     No_of_hours_in_Sleeping    0
     dtype: int64
```

```
df.shape
```

```
     (18942, 6)
```

```
gender_mapping = {1: 'Male', 2: 'Female'}

# Update the 'gender_column' using the map function
df['Gender'] = df['Gender'].map(gender_mapping)
```

```
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.countplot(x="Gender", hue='Trouble_Sleep', data=df)
plt.show()
```



```
gender_distribution = df['Gender'].value_counts().reset_index()
gender_distribution.columns = ['Gender', 'Count']
print("\nGender Distribution:")
print(gender_distribution.to_string(index=False))

# Construct frequency distribution for Age
age_distribution = df['Age '].value_counts().reset_index()
age_distribution.columns = ['Age', 'Count']
print("\nAge Distribution:")
print(age_distribution.to_string(index=False))

# Construct frequency distribution for Number of Hours Spent Sleeping
hours_sleeping_distribution = df['No_of_hours_in_Sleeping'].value_counts().reset_index()
hours_sleeping_distribution.columns = ['Hours_Sleeping', 'Count']
print("\nHours Sleeping Distribution:")
print(hours_sleeping_distribution.to_string(index=False))

# Construct frequency distribution for Frequency of Trouble Sleeping
trouble_sleeping_distribution = df['Trouble_Sleep'].value_counts().reset_index()
trouble_sleeping_distribution.columns = ['Trouble_Sleeping', 'Count']
print("\nTrouble Sleeping Distribution:")
print(trouble_sleeping_distribution.to_string(index=False))

# Construct frequency distribution for Frequency of Refreshing Sleep
refreshing_sleep_distribution = df['Refreshing_sleep'].value_counts().reset_index()
refreshing_sleep_distribution.columns = ['Refreshing_Sleep', 'Count']
print("\nRefreshing Sleep Distribution:")
print(refreshing_sleep_distribution.to_string(index=False))

# Construct frequency distribution for Frequency of Trouble Stay Awake
trouble_stay_awake_distribution = df['Trouble_Stay_awake'].value_counts().reset_index()
trouble_stay_awake_distribution.columns = ['Trouble_Stay_Awake', 'Count']
print("\nTrouble Stay Awake Distribution:")
print(trouble_stay_awake_distribution.to_string(index=False))
```

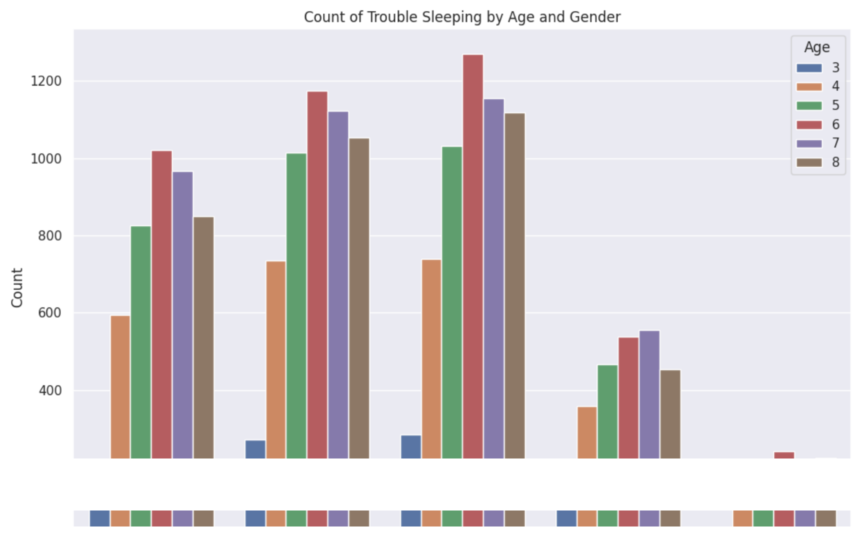
```
Gender Distribution:
Gender  Count
Female  10059
  Male   8883

Age Distribution:
 Age  Count
   6   4244
   7   4012
   8   3699
   5   3521
   4   2562
   3    904

Hours Sleeping Distribution:
 Hours_Sleeping  Count
              6   6488
              5   5100
              7   3701
              4   1968
              3    670
              8    559
              2    188
              9    177
              1     59
             10     32

Trouble Sleeping Distribution:
 Trouble_Sleeping  Count
                3   5599
```

```
                 2   5373
                 1   4449
                 4   2483
                 5   1038

Refreshing Sleep Distribution:
 Refreshing_Sleep  Count
                4   8508
                3   4978
                5   2382
                2   2279
                1    795

Trouble Stay Awake Distribution:
 Trouble_Stay_Awake  Count
                  1   7432
                  2   6043
                  3   4210
                  4    951
                  5    306
```

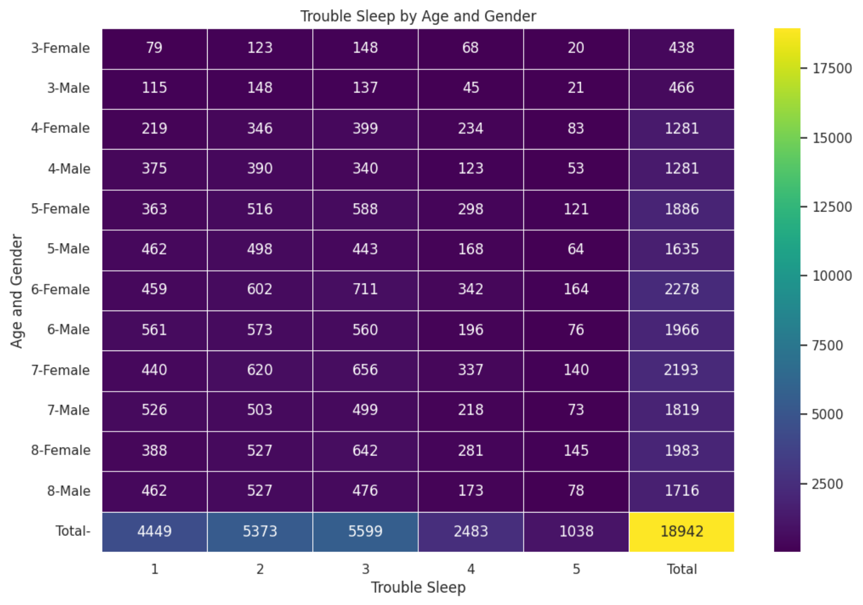
```
trouble_sleep_cross_tab = pd.crosstab(index=[df['Age '], df['Gender']], columns=df['Trouble_Sleep'], margins=True, margins_name="Total")

# Display the cross-tabulation
print(trouble_sleep_cross_tab)
```

```
Trouble_Sleep     1     2     3     4     5  Total
Age   Gender
3     Female     79   123   148    68    20    438
      Male      115   148   137    45    21    466
4     Female    219   346   399   234    83   1281
      Male      375   390   340   123    53   1281
5     Female    363   516   588   298   121   1886
      Male      462   498   443   168    64   1635
6     Female    459   602   711   342   164   2278
      Male      561   573   560   196    76   1966
7     Female    440   620   656   337   140   2193
      Male      526   503   499   218    73   1819
8     Female    388   527   642   281   145   1983
      Male      462   527   476   173    78   1716
Total          4449  5373  5599  2483  1038  18942
```

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



# Plotting
plt.figure(figsize=(12, 8))
sns.countplot(data=df, x='Trouble_Sleep', hue='Age ')
plt.title('Count of Trouble Sleeping by Age and Gender')
plt.xlabel('Trouble Sleeping')
plt.ylabel('Count')
plt.legend(title='Age')

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

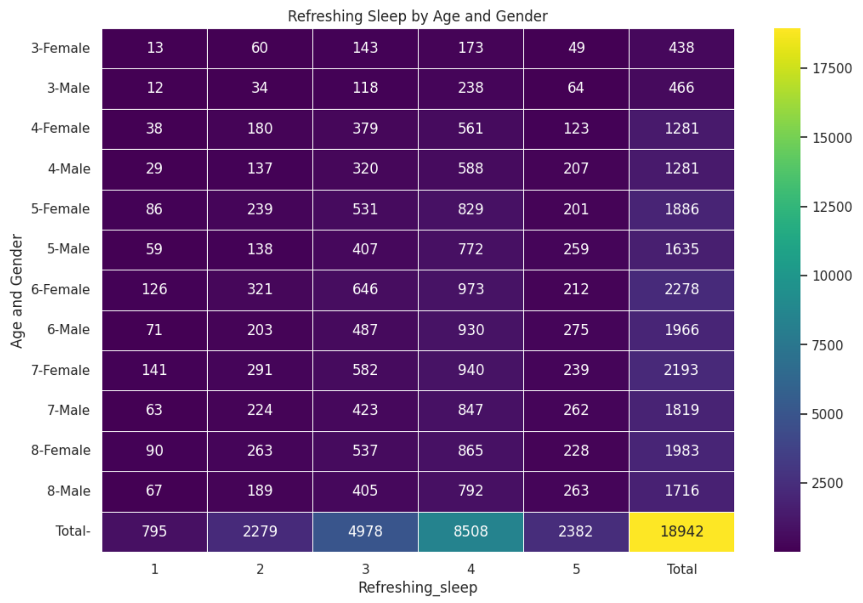


```
import matplotlib.pyplot as plt
import seaborn as sns



# Create a cross-tabulation
cross_tab = pd.crosstab(index=[df['Age '], df['Gender']], columns=df['Trouble_Sleep'], margins=True, margins_name="Total")

# Plotting
plt.figure(figsize=(12, 8))
sns.heatmap(cross_tab, annot=True, fmt='g', cmap='viridis', cbar=True, linewidths=.5)
plt.title('Trouble Sleep by Age and Gender')
plt.xlabel('Trouble Sleep')
plt.ylabel('Age and Gender')

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



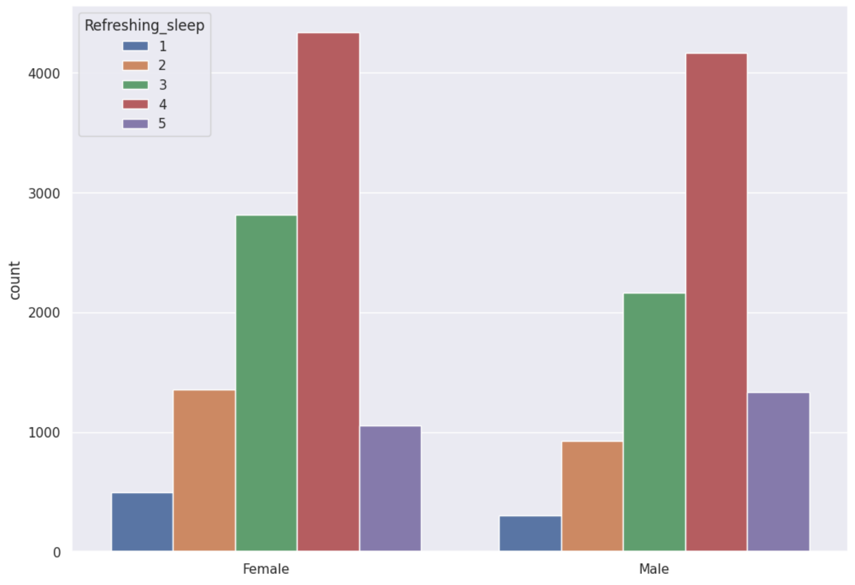
	1	2	3	4	5	Total
3-Female	79	123	148	68	20	438
3-Male	115	148	137	45	21	466
4-Female	219	346	399	234	83	1281
4-Male	375	390	340	123	53	1281
5-Female	363	516	588	298	121	1886
5-Male	462	498	443	168	64	1635
6-Female	459	602	711	342	164	2278
6-Male	561	573	560	196	76	1966
7-Female	440	620	656	337	140	2193
7-Male	526	503	499	218	73	1819
8-Female	388	527	642	281	145	1983
8-Male	462	527	476	173	78	1716
Total-	4449	5373	5599	2483	1038	18942

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



# Create a cross-tabulation
cross_tab = pd.crosstab(index=[df['Age '], df['Gender']], columns=df['Refreshing_sleep'], margins=True, margins_name="Total")

# Plotting
plt.figure(figsize=(12, 8))
sns.heatmap(cross_tab, annot=True, fmt='g', cmap='viridis', cbar=True, linewidths=.5)
plt.title('Refreshing Sleep by Age and Gender')
plt.xlabel('Refreshing_sleep')
plt.ylabel('Age and Gender')

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


Refreshing Sleep by Age and Gender

Age and Gender	1	2	3	4	5	Total
3-Female	13	60	143	173	49	438
3-Male	12	34	118	238	64	466
4-Female	38	180	379	561	123	1281
4-Male	29	137	320	588	207	1281
5-Female	86	239	531	829	201	1886
5-Male	59	138	407	772	259	1635
6-Female	126	321	646	973	212	2278
6-Male	71	203	487	930	275	1966
7-Female	141	291	582	940	239	2193
7-Male	63	224	423	847	262	1819
8-Female	90	263	537	865	228	1983
8-Male	67	189	405	792	263	1716
Total-	795	2279	4978	8508	2382	18942

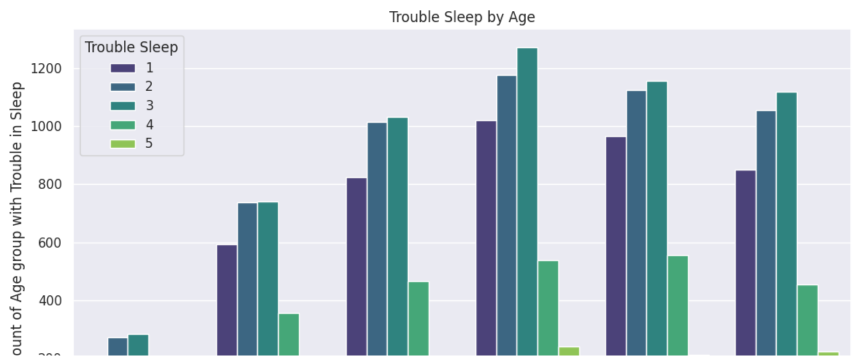
```
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.countplot(x="Gender", hue='Refreshing_sleep', data=df)
plt.show()
```



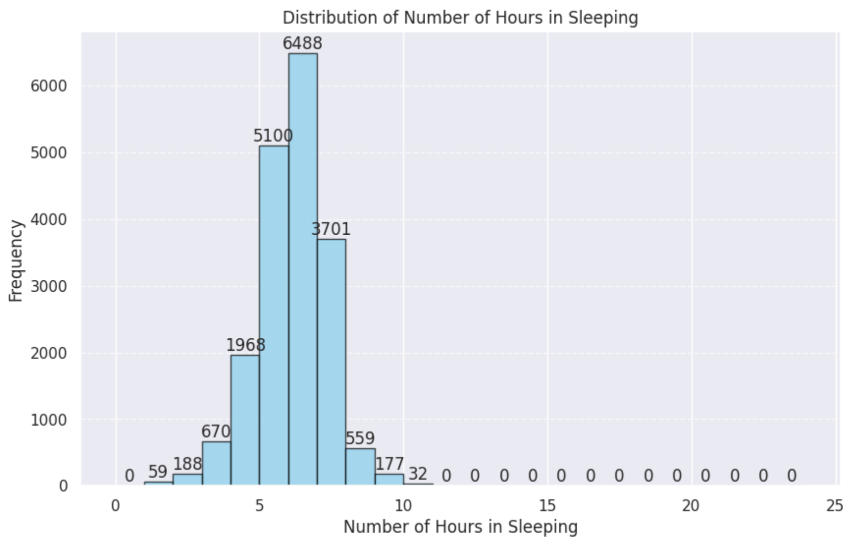
```
import matplotlib.pyplot as plt
import seaborn as sns



# Create a cross-tabulation
cross_tab = pd.crosstab(index=[df['Age '], df['Gender']], columns=df['Trouble_Sleep'], margins=True, margins_name="Total")

# Calculate the percentage values
cross_tab_percent = cross_tab.div(cross_tab['Total'], axis=0) * 100

# Plotting
plt.figure(figsize=(12, 8))
sns.heatmap(cross_tab_percent, annot=True, fmt='.2f', cmap='viridis', cbar=True, linewidths=.5)
plt.title('Trouble Sleep by Age and Gender (Percentage)')
plt.xlabel('Trouble Sleep')
plt.ylabel('Age and Gen')
```

```
Text(116.24999999999999, 0.5, 'Age and Gen')
```

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

# Assuming 'df' is your DataFrame with the required variables

# Plot between Age and Trouble_Sleep
plt.figure(figsize=(12, 6))
sns.countplot(data=df, x='Age ', hue='Trouble_Sleep',  palette='viridis')
plt.title('Trouble Sleep by Age')
plt.xlabel('Age Group')
plt.ylabel('Count of Age group with Trouble in Sleep')
plt.legend(title='Trouble Sleep')

# Show the plot
plt.show()

# Plot between Gender and Trouble_Sleep
plt.figure(figsize=(8, 6))
sns.countplot(data=df, x='Gender', hue='Trouble_Sleep', palette='viridis')
plt.title('Trouble Sleep by Gender')
plt.xlabel('Gender')
plt.ylabel('Count of Gender having trouble of sleep')
plt.legend(title='Trouble Sleep')

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

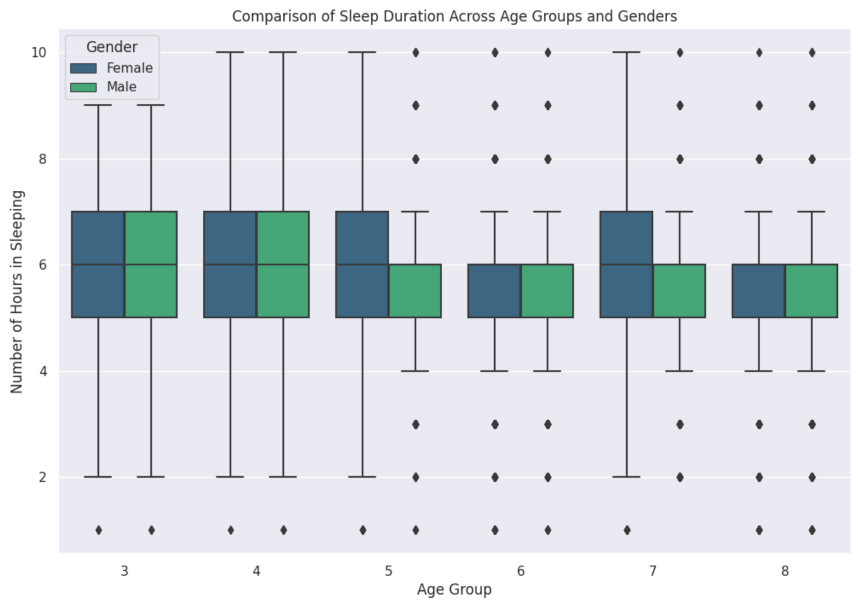


```
import matplotlib.pyplot as plt


# Plot histogram
plt.figure(figsize=(10, 6))
n, bins, patches = plt.hist(df['No_of_hours_in_Sleeping'], bins=range(0, 25), color='skyblue', edgecolor='black', alpha=0.7)
plt.title('Distribution of Number of Hours in Sleeping')
plt.xlabel('Number of Hours in Sleeping')
plt.ylabel('Frequency')
plt.grid(axis='y', linestyle='--', alpha=0.7)

# Annotate the frequency values on each bar
for bin_val, frequency in zip(bins, n):
    plt.text(bin_val + 0.5, frequency, str(int(frequency)), ha='center', va='bottom')

plt.show()

# Calculate summary statistics
summary_stats = df['No_of_hours_in_Sleeping'].describe()
print("Summary Statistics for Number of Hours in Sleeping:\n", summary_stats)
```

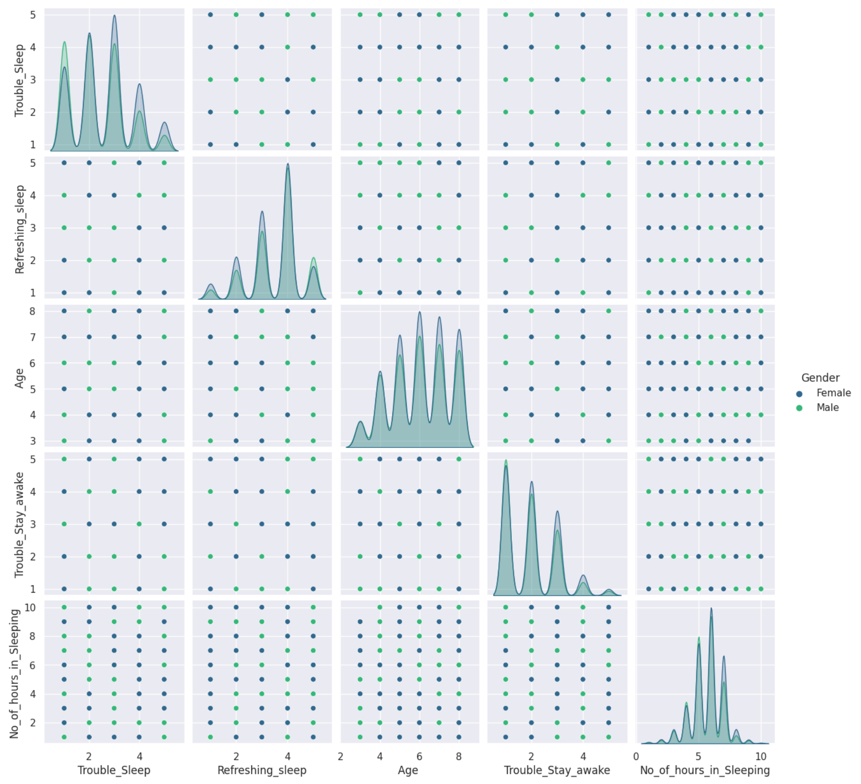


```
Summary Statistics for Number of Hours in Sleeping:
 count    18942.000000
mean         5.650776
std          1.241983
min          1.000000
25%          5.000000
50%          6.000000
75%          6.000000
max         10.000000
Name: No_of_hours_in_Sleeping, dtype: float64
```

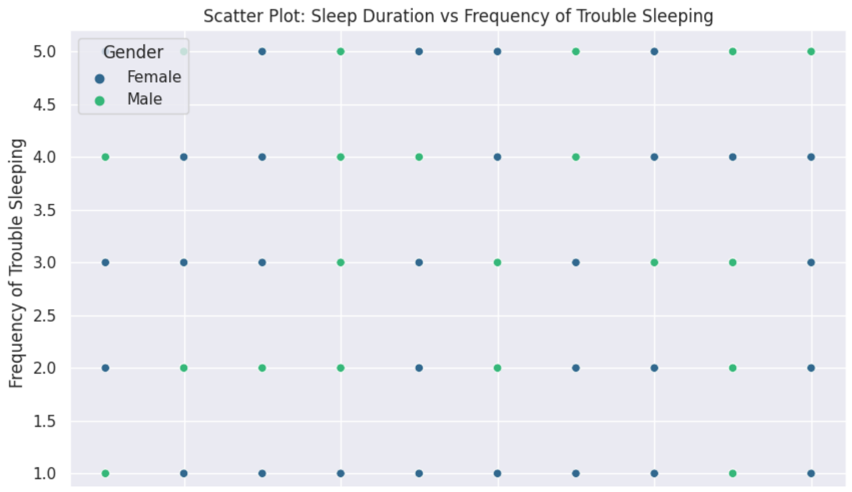
```
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.pyplot as plt

# Assuming 'df' is your DataFrame with sleep-related variables
# 'Age' is the column representing age groups, and 'No_of_hours_in_Sleeping' is sleep duration

# Set up the plot
plt.figure(figsize=(12, 8))
sns.boxplot(data=df, x='Age ', y='No_of_hours_in_Sleeping', hue='Gender', palette='viridis')

# Add labels and title
plt.title('Comparison of Sleep Duration Across Age Groups and Genders')
plt.xlabel('Age Group')
plt.ylabel('Number of Hours in Sleeping')

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



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

# Assuming 'df' is your DataFrame with sleep-related variables
# You can include multiple sleep-related variables in the pair plot

# Set up the plot
sns.pairplot(data=df, hue='Gender', palette='viridis')

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



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


# Set up the plot
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='No_of_hours_in_Sleeping', y='Refreshing_sleep', hue='Gender', palette='viridis')

# Add labels and title
plt.title('Scatter Plot: Sleep Duration vs Frequency of Trouble Sleeping')
plt.xlabel('Number of Hours in Sleeping')
plt.ylabel('Frequency of Trouble Sleeping')

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



```
# Define the criteria for sleep deprivation
sleep_deprivation_criteria = (df['Trouble_Sleep'] == 'Most of the time') & (df['No_of_hours_in_Sleeping'] < 6)

# Filter the DataFrame based on the criteria
sleep_deprivation_data = df[sleep_deprivation_criteria]

# Display the individuals experiencing sleep deprivation
print("Individuals experiencing sleep deprivation:")
print(sleep_deprivation_data)
```

```
Individuals experiencing sleep deprivation:
Empty DataFrame
Columns: [Gender, Trouble_Sleep, Refreshing_sleep, Age , Trouble_Stay_awake, No_of_hours_in_Sleeping]
Index: []
```

```
# Create a new column 'Meets_Sleep_Guidelines' based on age
df['Meets_Sleep_Guidelines'] = (
    ((df['Age '] >= 3) & (df['Age '] <= 4) & (df['No_of_hours_in_Sleeping'] >= 7) & (df['No_of_hours_in_Sleeping'] <= 9)) |
    ((df['Age '] >= 5) & (df['No_of_hours_in_Sleeping'] >= 7) & (df['No_of_hours_in_Sleeping'] <= 9))
)

# Calculate the percentage of individuals meeting sleep duration guidelines
percentage_meeting_guidelines = (df['Meets_Sleep_Guidelines'].sum() / len(df)) * 100

print(f"{percentage_meeting_guidelines:.2f}% of individuals meet the recommended sleep duration guidelines.")
```

```
23.42% of individuals meet the recommended sleep duration guidelines.
```

```
# Check unique values in the 'Trouble_Sleep' column
unique_values = df['Trouble_Sleep'].unique()

# Verify that 'Sometimes' and 'Most of the time' are valid categories
valid_categories = [3,4,5]
if not set(valid_categories).issubset(set(unique_values)):
    print("Invalid categories in 'Trouble_Sleep' column.")
else:
    # Calculate the percentage
    total_individuals = len(df)
    trouble_sleep_count = df['Trouble_Sleep'].isin(valid_categories).sum()
    percentage_trouble_sleep = (trouble_sleep_count / total_individuals) * 100

    # Print the result
    print(f"Percentage of individuals with trouble sleeping: {percentage_trouble_sleep:.2f}%")
```

```
Percentage of individuals with trouble sleeping: 48.15%
```

```
# Calculate the percentage of men and women facing sleep quality challenges
total_men = df[df['Gender'] == 'Male'].shape[0]
total_women = df[df['Gender'] == 'Female'].shape[0]

men_sleep_challenges = df[(df['Gender'] == 'Male') & (df['Trouble_Sleep'].isin([3,4,5]))].shape[0]
women_sleep_challenges = df[(df['Gender'] == 'Female') & (df['Trouble_Sleep'].isin([3,4,5]))].shape[0]
```

```
percentage_men_challenges = (men_sleep_challenges / total_men) * 100
percentage_women_challenges = (women_sleep_challenges / total_women) * 100

print(f"Percentage of men facing sleep quality challenges: {percentage_men_challenges:.2f}%")
print(f"Percentage of women facing sleep quality challenges: {percentage_women_challenges:.2f}%")
```

```
Percentage of men facing sleep quality challenges: 42.14%
Percentage of women facing sleep quality challenges: 53.45%
```

```
!pip install pandoc
import pandoc
```

```
Collecting pandoc
  Downloading pandoc-2.3.tar.gz (33 kB)
  Preparing metadata (setup.py) ... done
Collecting plumbum (from pandoc)
  Downloading plumbum-1.8.2-py3-none-any.whl (127 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.0/127.0 kB 4.0 MB/s eta 0:00:00
Collecting ply (from pandoc)
  Downloading ply-3.11-py2.py3-none-any.whl (49 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.6/49.6 kB 6.1 MB/s eta 0:00:00
Building wheels for collected packages: pandoc
  Building wheel for pandoc (setup.py) ... done
  Created wheel for pandoc: filename=pandoc-2.3-py3-none-any.whl size=33261 sha256=87be07bb3b7affbb77cb3c702c093f3675ac05e9ab49a40d2
  Stored in directory: /root/.cache/pip/wheels/76/27/c2/c26175310aadcb8741b77657a1bb49c50cc7d4cdbf9eee0005
Successfully built pandoc
Installing collected packages: ply, plumbum, pandoc
Successfully installed pandoc-2.3 plumbum-1.8.2 ply-3.11
```