```
print("1.4.1 How can we determine the average daily social media usage time among young adults in the sample, and how does it compare to their overall scree
```

```
1.4.1 How can we determine the average daily social media usage time among young adults in the sample, and how does it compare to their overall screen
```

```
from google.colab import files
```

```
uploaded = files.upload()
```

```
Choose Files No file chosen  Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving mental_health_and_technology_usage_2022.xlsx to mental_health_and_technology_usage_2022 (1).xlsx
```

```
import pandas as pd
data = pd.read_excel('mental_health_and_technology_usage_2022.xlsx')
```

```
data
```

	Timestap	User_ID	Age	Birth Year	Generation	Technology_Usage_Hours	Social_Media_Usage_Hours	Gaming_Hours	Screen_Time_Hours	Mental_Health_Statu
0	2022-04-01	USER-00001	23	1999	Gen Z	6.57	6.00	0.68	12.36	Goo
1	2022-04-01	USER-00002	21	2001	Gen Z	3.01	2.57	3.74	7.61	Poc
2	2022-04-01	USER-00003	51	1971	Gen X	3.04	6.14	1.26	3.16	Fa
3	2022-04-01	USER-00004	25	1997	Gen Z	3.84	4.48	2.59	13.08	Exceller
4	2022-04-01	USER-00005	53	1969	Gen X	1.20	0.56	0.29	12.63	Goo
...	...	...	...	...	...	...	...	...	...	.
9995	2022-12-02	USER-09996	42	1980	Gen X	7.05	0.41	0.53	13.90	Goo
9996	2022-12-02	USER-09997	31	1991	Millennials	3.12	6.79	0.80	1.17	Fa
9997	2022-12-02	USER-09998	23	1999	Gen Z	4.38	3.98	0.52	7.81	Poc
9998	2022-12-02	USER-09999	38	1984	Millennials	4.44	1.48	3.28	13.95	Poc
9999	2022-12-02	USER-10000	41	1981	Millennials	2.50	4.80	0.25	8.82	Fa

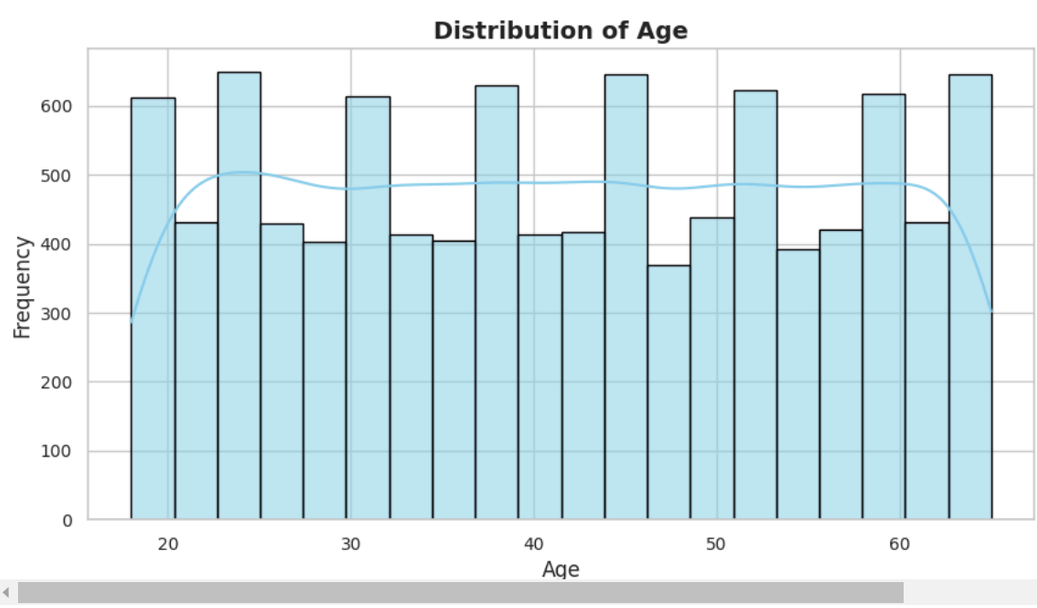
10000 rows × 16 columns

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 16 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Timestap                  10000 non-null  datetime64[ns]
 1   User_ID                   10000 non-null  object
 2   Age                       10000 non-null  int64
 3   Birth Year                10000 non-null  int64
 4   Generation                10000 non-null  object
 5   Technology_Usage_Hours    10000 non-null  float64
 6   Social_Media_Usage_Hours  10000 non-null  float64
 7   Gaming_Hours              10000 non-null  float64
 8   Screen_Time_Hours         10000 non-null  float64
 9   Mental_Health_Status      10000 non-null  object
 10  Stress_Level              10000 non-null  int64
 11  Sleep_Hours               10000 non-null  float64
 12  Physical_Activity_Hours   10000 non-null  float64
 13  Support_Systems_Access    10000 non-null  object
 14  Work_Environment_Impact   10000 non-null  object
 15  Online_Support_Usage      10000 non-null  object
dtypes: datetime64[ns](1), float64(6), int64(3), object(6)
memory usage: 1.2+ MB
```

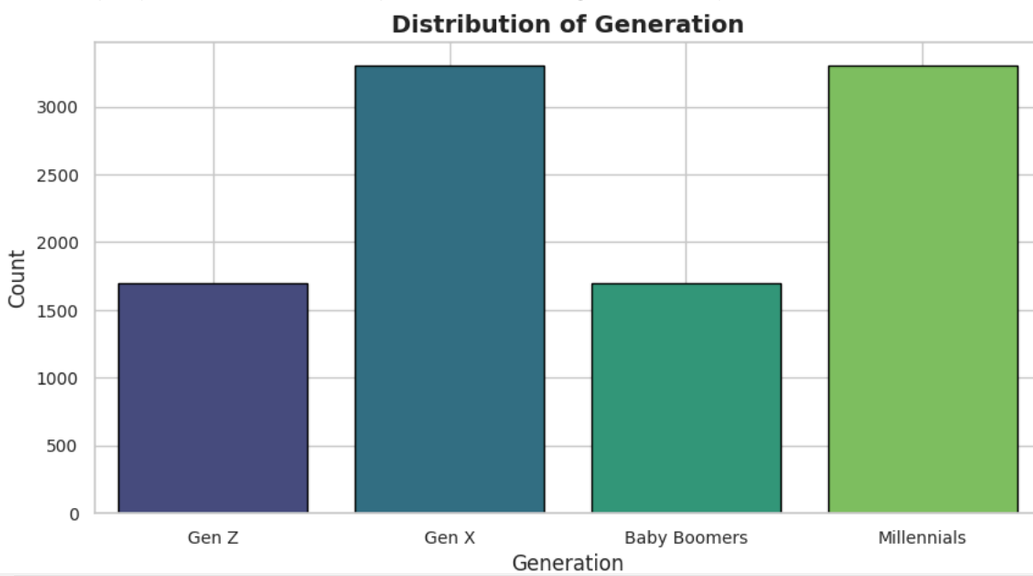
```
df = data
```

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

# Load the dataset (if not already loaded)
# df = pd.read_csv('path_to_your_dataset.csv')

# Set the plot style
sns.set(style="whitegrid")

# Distribution chart for Age (Histogram)
plt.figure(figsize=(10, 5))
sns.histplot(df['Age'], bins=20, kde=True, color='skyblue', edgecolor='black')
plt.title('Distribution of Age', fontsize=14, fontweight='bold')
plt.xlabel('Age', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.grid(True)
plt.show()
```




```
# Distribution chart for Generation (Bar Chart)
plt.figure(figsize=(10, 5))
sns.countplot(x='Generation', data=df, palette='viridis', edgecolor='black')

plt.xlabel('Generation', fontsize=12)
plt.ylabel('Count', fontsize=12)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.grid(True)
plt.show()
```

```
<ipython-input-44-5376213a0210>:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the

  sns.countplot(x='Generation', data=df, palette='viridis', edgecolor='black')
```

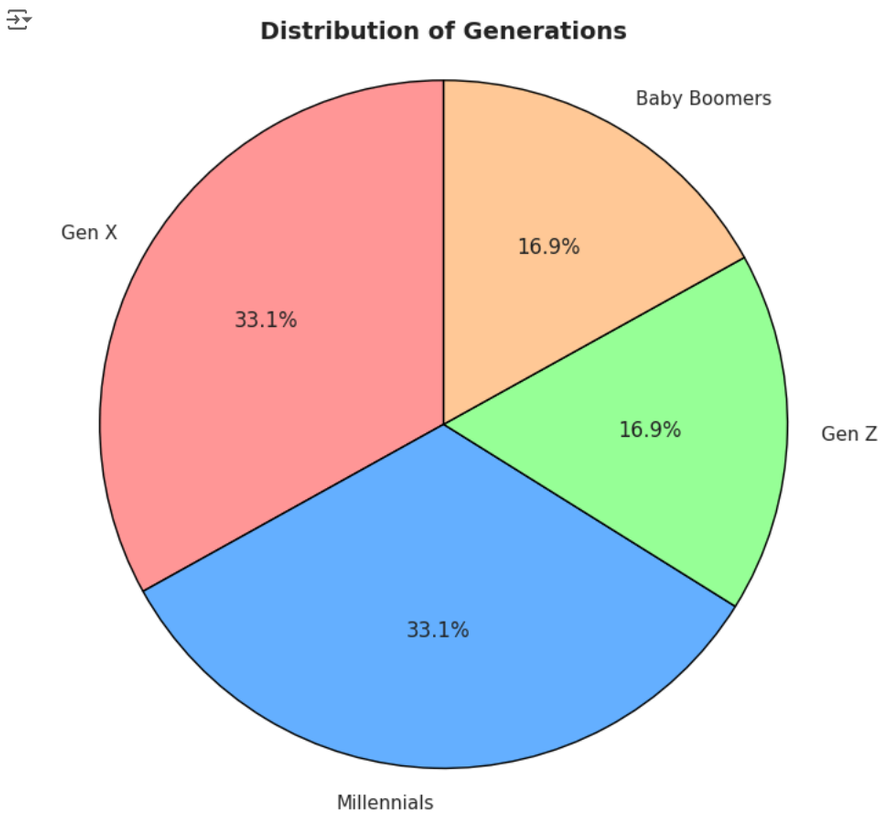



```
import matplotlib.pyplot as plt

# Calculate the distribution of the Generation column
generation_counts = df['Generation'].value_counts()

# Create a pie chart
plt.figure(figsize=(8, 8))  # Set figure size
plt.pie(generation_counts, labels=generation_counts.index, autopct='%1.1f%%', startangle=90, colors=['#ff9999','#66b3ff','#99ff99','#ffcc99','#c2c2f0'], ed

# Equal aspect ratio ensures that pie is drawn as a circle.
plt.axis('equal')

# Add a title
plt.title('Distribution of Generations', fontsize=14, fontweight='bold')

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

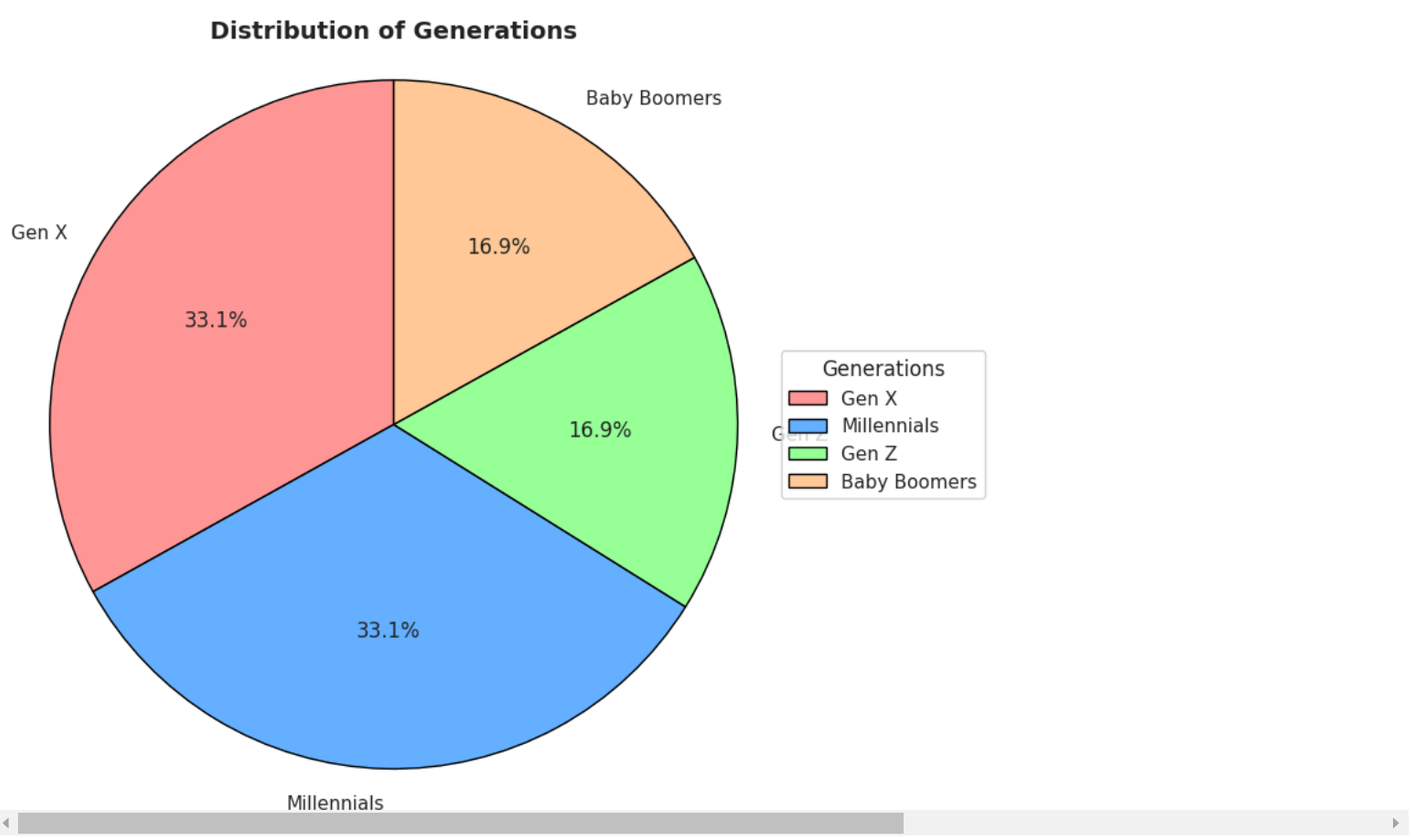
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-45-24c58080d992> in <cell line: 8>()
      6 # Create a pie chart
      7 plt.figure(figsize=(8, 8))  # Set figure size
----> 8 plt.pie(generation_counts, labels=generation_counts.index, autopct='%1.1f%%', startangle=90, colors=
['#ff9999','#66b3ff','#99ff99','#ffcc99','#c2c2f0'], edgecolor='black')
      9
     10 # Equal aspect ratio ensures that pie is drawn as a circle.

TypeError: pie() got an unexpected keyword argument 'edgecolor'
```

```
import matplotlib.pyplot as plt

# Calculate the distribution of the Generation column
```

```
generation_counts = df['Generation'].value_counts()

# Create a pie chart
plt.figure(figsize=(8, 8))  # Set figure size
wedges, texts, autotexts = plt.pie(generation_counts, labels=generation_counts.index, autopct='%1.1f%%', startangle=90, colors=['#ff9999','#66b3ff','#99ff99
#Get wedges for setting edgecolor

# Set the edgecolor for each wedge
for w in wedges:
    w.set_edgecolor('black')


# Equal aspect ratio ensures that pie is drawn as a circle.
plt.axis('equal')

# Add a title
plt.title('Distribution of Generations', fontsize=14, fontweight='bold')

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




```
import matplotlib.pyplot as plt

# Calculate the distribution of the Generation column
generation_counts = df['Generation'].value_counts()

# Define the colors for the pie chart
colors = ['#ff9999', '#66b3ff', '#99ff99', '#ffcc99', '#c2c2f0']

# Create a pie chart
plt.figure(figsize=(8, 8))  # Set figure size
wedges, texts, autotexts = plt.pie(generation_counts, labels=generation_counts.index, autopct='%1.1f%%', startangle=90, colors=colors)

# Set the edgecolor for each wedge
for w in wedges:
    w.set_edgecolor('black')

# Add a legend with the generation labels
plt.legend(wedges, generation_counts.index, title="Generations", loc="center left", bbox_to_anchor=(1, 0, 0.5, 1))

# Equal aspect ratio ensures that pie is drawn as a circle.
plt.axis('equal')

# Add a title
plt.title('Distribution of Generations', fontsize=14, fontweight='bold')

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

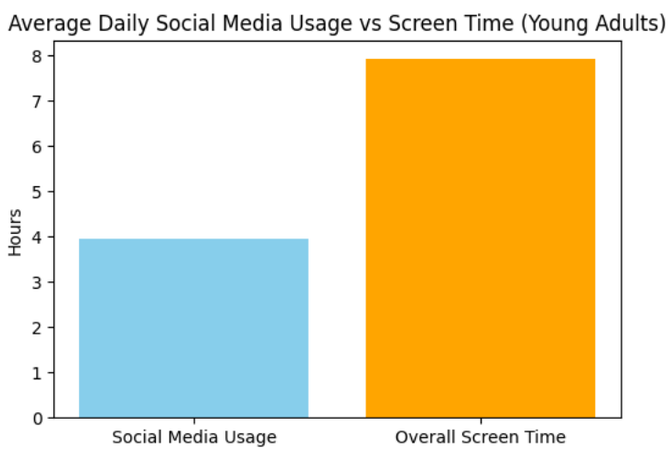



```
# Define young adults as those between 18 and 35 years old
young_adults = data[(data['Age'] >= 18) & (data['Age'] <= 35)]

# Calculate the average daily social media usage and screen time for young adults
average_social_media_usage = young_adults['Social_Media_Usage_Hours'].mean()
average_screen_time = young_adults['Screen_Time_Hours'].mean()

# Print the results
print(f"Average daily social media usage among young adults: {average_social_media_usage:.2f} hours")
print(f"Average overall screen time among young adults: {average_screen_time:.2f} hours")
```

```
Average daily social media usage among young adults: 3.95 hours
Average overall screen time among young adults: 7.93 hours
```

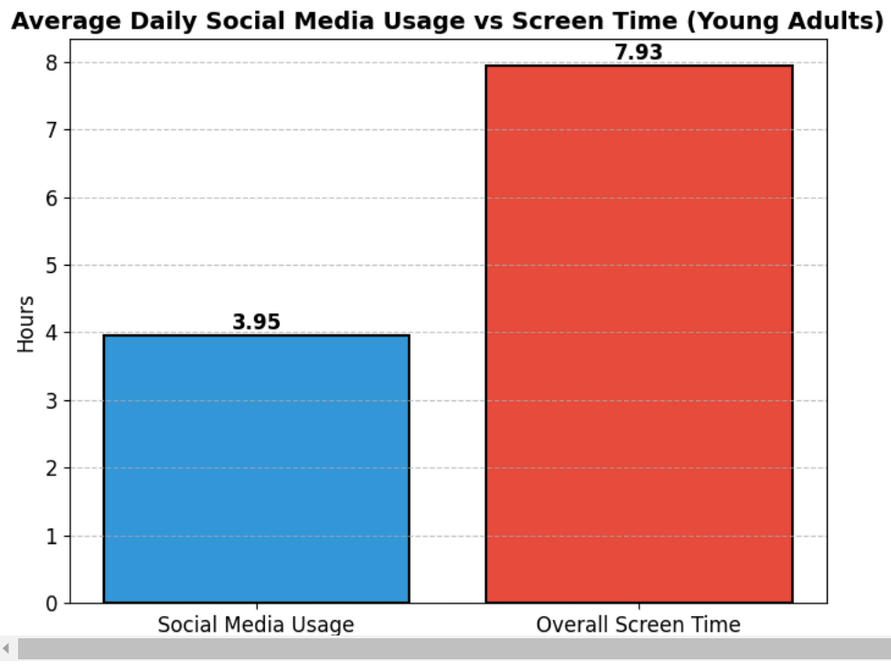
```
import matplotlib.pyplot as plt
import pandas as pd
```

```
# Plotting the results
labels = ['Social Media Usage', 'Overall Screen Time']
averages = [average_social_media_usage, average_screen_time]

plt.figure(figsize=(6, 4))
plt.bar(labels, averages, color=['skyblue', 'orange'])
plt.title('Average Daily Social Media Usage vs Screen Time (Young Adults)')
plt.ylabel('Hours')
plt.show()
```




```
labels = ['Social Media Usage', 'Overall Screen Time']
averages = [average_social_media_usage, average_screen_time]

plt.figure(figsize=(8, 6))  # Larger figure size for clarity
bars = plt.bar(labels, averages, color=['#3498db', '#e74c3c'], edgecolor='black', linewidth=1.5)

# Adding values on top of bars
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval + 0.1, round(yval, 2), ha='center', fontsize=12, fontweight='bold')

# Enhance the grid and axes
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.title('Average Daily Social Media Usage vs Screen Time (Young Adults)', fontsize=14, fontweight='bold')
plt.ylabel('Hours', fontsize=12)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
```

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



1.4.6 How do technology usage patterns and stress levels vary across generations (The Silent Generation, Baby Boomers, Gen X, Millennials, Gen Z, and Gen Alpha)?

Algorithms: Data Aggregation, Summary Statistics

```
stress_mapping = {
    'Low': 1,
    'Medium': 2,
    'High': 3
}

# Convert 'Stress_Level' to numeric values
data['Stress_Level'] = data['Stress_Level'].map(stress_mapping)

# Check if 'Technology_Usage_Hours' contains any non-numeric values
# If so, try converting them to numeric, forcing errors to NaN
data['Technology_Usage_Hours'] = pd.to_numeric(data['Technology_Usage_Hours'], errors='coerce')

# Drop rows where either column has missing values after conversion
data_cleaned = data.dropna(subset=['Technology_Usage_Hours', 'Stress_Level'])

# Select relevant columns for analysis
relevant_columns = ['Generation', 'Technology_Usage_Hours', 'Stress_Level']
data_subset = data_cleaned[relevant_columns]

# Group by 'Generation' and calculate summary statistics (mean and standard deviation)
grouped_data = data_subset.groupby('Generation').agg(
    avg_tech_usage=('Technology_Usage_Hours', 'mean'),
    std_tech_usage=('Technology_Usage_Hours', 'std'),
    avg_stress_level=('Stress_Level', 'mean'),
    std_stress_level=('Stress_Level', 'std')
).reset_index()

# Display the summary statistics for each generation
print(grouped_data)
```

```
      Generation  avg_tech_usage  std_tech_usage  avg_stress_level  \
0  Baby Boomers        6.561423        3.149514          1.976387
1         Gen X        6.513269        3.176346          2.013003
2         Gen Z        6.460697        3.180053          2.024793
3   Millennials        6.397749        3.165632          1.985779

   std_stress_level
0          0.815189
1          0.818428
2          0.817325
3          0.813835
```

```
pip install tabulate
```

```
Requirement already satisfied: tabulate in /usr/local/lib/python3.10/dist-packages (0.9.0)
```

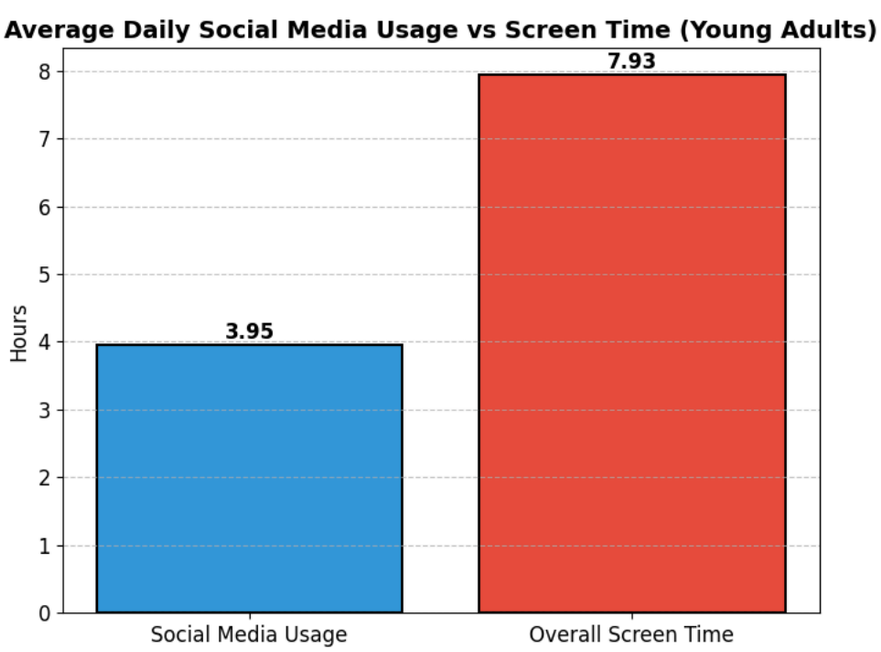
```
from tabulate import tabulate
```

```
# Pretty print the summary statistics using the tabulate library for a clean table format
table = tabulate(grouped_data, headers='keys', tablefmt='fancy_grid', showindex=False)

# Print the table
print(table)
```

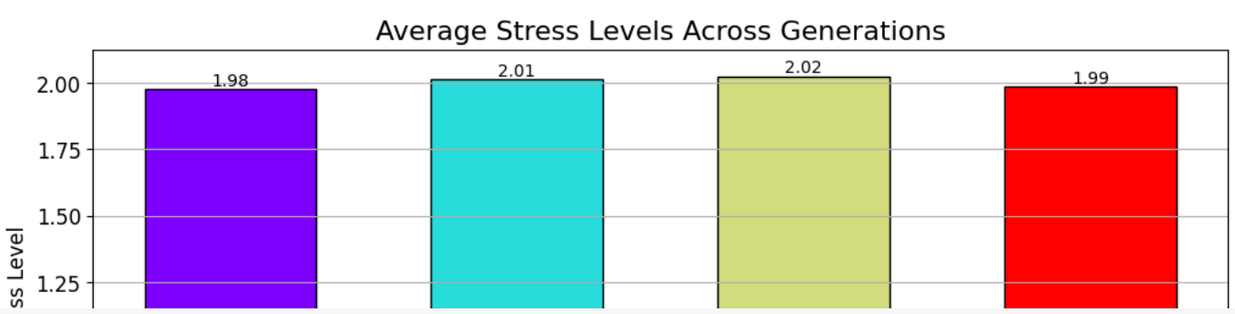
```
╒═══════════════╤═══════════════════╤═══════════════════╤═══════════════════╤═══════════════════╕
│ Generation    │   avg_tech_usage  │   std_tech_usage  │   avg_stress_level  │   std_stress_level  │
╞═══════════════╪═══════════════════╪═══════════════════╪═══════════════════╪═══════════════════╡
│ Baby Boomers  │           6.56142 │           3.14951 │           1.97639 │          0.815189 │
├───────────────┼───────────────────┼───────────────────┼───────────────────┼───────────────────┤
│ Gen X         │           6.51327 │           3.17635 │           2.013   │          0.818428 │
├───────────────┼───────────────────┼───────────────────┼───────────────────┼───────────────────┤
│ Gen Z         │           6.4607  │           3.18005 │           2.02479 │          0.817325 │
```

Millennials	6.39775	3.16563	1.98578	0.813835

```
import matplotlib.pyplot as plt
from google.colab import files  # Only needed in Google Colab

# Define the labels and average values
labels = ['Social Media Usage', 'Overall Screen Time']
averages = [average_social_media_usage, average_screen_time]

# Create the figure with larger size
plt.figure(figsize=(8, 6))

# Create the bar chart with custom colors and edge
bars = plt.bar(labels, averages, color=['#3498db', '#e74c3c'], edgecolor='black', linewidth=1.5)

# Adding values on top of the bars
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval + 0.1, round(yval, 2), ha='center', fontsize=12, fontweight='bold')

# Customize the grid and axes
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.title('Average Daily Social Media Usage vs Screen Time (Young Adults)', fontsize=14, fontweight='bold')
plt.ylabel('Hours', fontsize=12)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)

# Save the chart as a PNG image before showing it
plt.savefig('avg_levels_chart.png')

# Show the plot
plt.show()

# Download the image (specific to Google Colab)
files.download('avg_levels_chart.png')
```




```
# Bar chart for average stress levels across generations
plt.figure(figsize=(12, 6))
bars = plt.bar(grouped_data['Generation'], grouped_data['avg_stress_level'], color=colors_stress, edgecolor='black', width=0.6)
plt.title('Average Stress Levels Across Generations', fontsize=16)
plt.xlabel('Generation', fontsize=12)
plt.ylabel('Average Stress Level', fontsize=12)
plt.grid(True, axis='y')

# Add data labels on top of the bars
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval, round(yval, 2), va='bottom', ha='center', fontsize=10)

plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.show()
plt.savefig('stress_levels_chart.png') # Save the chart as a PNG image
files.download('stress_levels_chart.png') # Download the saved image
```




```
import matplotlib.pyplot as plt
from google.colab import files  # Only needed in Google Colab

# Bar chart for average stress levels across generations (horizontal)
plt.figure(figsize=(12, 6))

# Create horizontal bar chart
bars = plt.barh(grouped_data['Generation'], grouped_data['avg_stress_level'], color=colors_stress, edgecolor='black', height=0.6)

# Title and labels
plt.title('Average Stress Levels Across Generations', fontsize=16)
plt.xlabel('Average Stress Level', fontsize=12)
plt.ylabel('Generation', fontsize=12)
plt.grid(True, axis='x')

# Add data labels next to the bars
for bar in bars:
    xval = bar.get_width()
    plt.text(xval, bar.get_y() + bar.get_height()/2, round(xval, 2), va='center', ha='left', fontsize=10)

# Customize ticks
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)

# Save the chart as a PNG image before showing it
plt.savefig('stress_levels_chart.png')

# Show the plot
plt.show()

# Download the image (specific to Google Colab)
files.download('stress_levels_chart.png')
```