Project Report: Healthy Lifestyle - A detailed analysis of health application trends

1. Problem Statement:

Introduction:

Leading a healthy lifestyle is a multifaceted challenge that involves various factors such as diet, physical activity, sleep patterns, and mental well-being. Understanding and analyzing these factors is crucial for developing personalized recommendations and strategies that promote overall well-being. In this project, we aim to use data analytics to identify key patterns and correlations within lifestyle factors, providing actionable insights for individuals seeking to improve their health.

Goal:

Develop a data analytics framework using Python to analyze and gain insights into various aspects of a healthy lifestyle, ultimately aiding individuals in making informed decisions about their well-being.

Objectives:

1. Data Acquisition:

Collect a comprehensive dataset encompassing various lifestyle factors such as diet, exercise, sleep patterns, and mental health indicators. Ensure the dataset is clean, well-labeled, and formatted for analysis.

2. Data Analysis:

Perform exploratory data analysis (EDA) to uncover trends, correlations, and patterns within the lifestyle dataset. Investigate how different factors interact and contribute to overall health and well-being.

3. Data Preprocessing:

Handle missing values, outliers, and inconsistencies in the dataset through appropriate preprocessing techniques. Normalize or scale numerical features, encode categorical variables, and split the data into training and testing sets as needed.

4. Feature Engineering:

Derive new features or metrics that provide deeper insights into the lifestyle data. For example, calculate the average daily physical activity level or derive a sleep quality index based on sleep duration and disturbances.

5. **Model Selection:**

Evaluate and compare various machine learning models and statistical techniques to analyze and predict health outcomes based on lifestyle data. Consider models such as logistic regression, decision trees, random forests, and clustering algorithms for this purpose.

6. Model Training:

Train the selected models on the processed dataset, enabling them to learn from the underlying patterns and relationships within the data. Fine-tune model hyperparameters to optimize performance.

7. Model Evaluation:

Assess the performance of the trained models using appropriate evaluation metrics such as accuracy, precision, recall, F1-score, and cluster purity (if applicable). Evaluate the models' ability to provide actionable insights and predictions regarding healthy lifestyle choices.

2. Data Description:

The dataset for this project includes various lifestyle factors such as dietary habits, physical activity levels, sleep patterns, and mental health indicators. Data is collected from reliable sources such as health surveys, fitness trackers, and self-reported logs. The dataset is preprocessed to ensure it is clean, complete, and consistent, ready for analysis and model training.

3. Solution Approach:

1. Logistic Regression:

-Working:

Logistic regression can be used to predict binary outcomes, such as whether a person maintains a healthy lifestyle or not, based on various factors such as diet, exercise, and sleep patterns. The model estimates the probability of an outcome by fitting a logistic curve to the data.

- Uses:

Useful in identifying key lifestyle factors that significantly impact health outcomes, such as the likelihood of developing lifestyle-related diseases.

2. Decision Tree:

- Working:

The decision tree algorithm works by recursively splitting the dataset based on the most significant features, creating a tree-like structure. Each node represents a decision rule that leads to an outcome at the leaf nodes.

- Uses:

Decision trees can identify the most critical lifestyle factors influencing health, providing clear and interpretable rules for making healthier choices.

3. Random Forest:

- Working:

Random Forest builds an ensemble of decision trees to make predictions. Each tree is trained on a random subset of the data, and the final prediction is made by averaging the predictions from all trees, reducing the risk of overfitting.

- Uses:

Random Forest can help in predicting overall health status by considering various lifestyle factors simultaneously, providing robust and accurate insights.

4. Clustering Algorithms (e.g., K-Means):

- Working:

Clustering algorithms group similar data points together based on feature similarity. For example, K-Means clustering assigns data points into distinct clusters, helping to identify different lifestyle patterns among individuals.

- Uses:

Useful for segmenting individuals into groups based on their lifestyle habits, which can then be used to tailor personalized health recommendations.

5. Support Vector Machine (SVM):

- Working:

SVM finds the optimal hyperplane that separates data points into different classes, such as healthy vs. unhealthy lifestyle choices. It maximizes the margin between classes, making it effective for binary classification.

- Uses:

SVM can be applied to classify lifestyle choices and predict health outcomes, particularly in scenarios with complex decision boundaries.

6. Principal Component Analysis (PCA):

- Working:

PCA is a dimensionality reduction technique that transforms the dataset into a set of linearly uncorrelated components. It helps in reducing the complexity of the dataset while retaining the most important information.

- Uses:

PCA can be used to identify the most influential factors in lifestyle data, simplifying the analysis and highlighting the key determinants of a healthy lifestyle.

4. Planning:

Days 1-3 (Defining Scope and Dataset Identification):

- Define the scope of the project, including the objectives and deliverables for analyzing healthy lifestyle data.
- Identify and gather relevant datasets, such as dietary logs, physical activity records, sleep patterns, and mental health surveys.
- Create basic functionalities to preprocess and explore the dataset, such as data cleaning and visualization tools.

Days 4-7 (Dataset Analysis and Feature Engineering):

- Analyze the collected dataset to understand its structure, distribution, and characteristics.
- Identify key features and patterns that influence health outcomes, such as diet quality, exercise frequency, and sleep duration.
- Engineer new features or metrics that provide deeper insights into the lifestyle data.

Days 8-10 (Model Training and Testing):

- Train selected machine learning models on the dataset to analyze and predict health outcomes based on lifestyle factors.
- Test the performance of the models, gathering feedback for adjustments and improvements.

Days 11-15 (System Enhancement, Deployment, and Release):

- Enhance the system based on feedback and scalability requirements, making necessary adjustments to improve performance.
- Deploy the predictive models in a controlled environment using tools like Streamlit or similar platforms.

- Perform end-to-end testing of the deployed system to ensure reliability and accuracy.

- Prepare for the release of the system, providing documentation and support for implementation and usage.

5. Dataset Overview:

- Date: Date of data entry or observation.

- Diet: Details on daily dietary intake (e.g., calories, macronutrients).

- Exercise: Information on daily physical activity (e.g., steps, exercise duration).

- Sleep: Data on sleep duration and quality.

- Mental Health: Self-reported mental well-being scores or stress levels.

- Health Metrics: Objective health indicators such as weight, BMI, and blood pressure.

```
import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
%matplotlib inline

import os
# os.listdir('../input')
# Check matplotlib version
print(matplotlib.__version__)
print(pd.__version__)
print(np.__version__)
```

```
3.8.0
2.1.4
1.26.3
```

```
df = pd.read_csv('../dataset/Wellbeing_and_lifestyle_data_Kaggle.csv')
df.head()
```

```
  Timestamp  FRUITS_VEGGIES DAILY_STRESS  PLACES_VISITED  CORE_CIRCLE  
\
0    7/7/15               3            2               2            5

1    7/7/15               2            3               4            3

2    7/7/15               2            3               3            4

3    7/7/15               3            3              10            3

4    7/7/15               5            1               3            3


   SUPPORTING_OTHERS  SOCIAL_NETWORK  ACHIEVEMENT  DONATION  BMI_RANGE 
...  \
0                  0               5            2         0          1 
...
1                  8              10            5         2          2 
...
2                  4              10            3         2          2 
...
3                 10               7            2         5          2 
...
4                 10               4            2         4          2 
...

   SLEEP_HOURS  LOST_VACATION  DAILY_SHOUTING  SUFFICIENT_INCOME  \
0            7              5               5                  1
1            8              2               2                  2
```

```
2          8          10          2          2
3          5           7          5          1
4          7           0          0          2

   PERSONAL_AWARDS  TIME_FOR_PASSION  WEEKLY_MEDITATION        AGE
GENDER  \
0                4                 0                  5    36 to 50
Female
1                3                 2                  6    36 to 50
Female
2                4                 8                  3    36 to 50
Female
3                5                 2                  0  51 or more
Female
4                8                 1                  5  51 or more
Female

   WORK_LIFE_BALANCE_SCORE
0                    609.5
1                    655.6
2                    631.6
3                    622.7
4                    663.9

[5 rows x 24 columns]
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15972 entries, 0 to 15971
Data columns (total 24 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Timestamp                15972 non-null  object
 1   FRUITS_VEGGIES           15972 non-null  int64
 2   DAILY_STRESS             15972 non-null  object
 3   PLACES_VISITED           15972 non-null  int64
 4   CORE_CIRCLE              15972 non-null  int64
 5   SUPPORTING_OTHERS        15972 non-null  int64
 6   SOCIAL_NETWORK           15972 non-null  int64
 7   ACHIEVEMENT              15972 non-null  int64
 8   DONATION                 15972 non-null  int64
 9   BMI_RANGE                15972 non-null  int64
 10  TODO_COMPLETED           15972 non-null  int64
 11  FLOW                     15972 non-null  int64
 12  DAILY_STEPS              15972 non-null  int64
 13  LIVE_VISION              15972 non-null  int64
 14  SLEEP_HOURS              15972 non-null  int64
 15  LOST_VACATION            15972 non-null  int64
 16  DAILY_SHOUTING           15972 non-null  int64
```

```
 17  SUFFICIENT_INCOME        15972 non-null  int64
 18  PERSONAL_AWARDS          15972 non-null  int64
 19  TIME_FOR_PASSION         15972 non-null  int64
 20  WEEKLY_MEDITATION        15972 non-null  int64
 21  AGE                      15972 non-null  object
 22  GENDER                   15972 non-null  object
 23  WORK_LIFE_BALANCE_SCORE  15972 non-null  float64
dtypes: float64(1), int64(19), object(4)
memory usage: 2.9+ MB
```

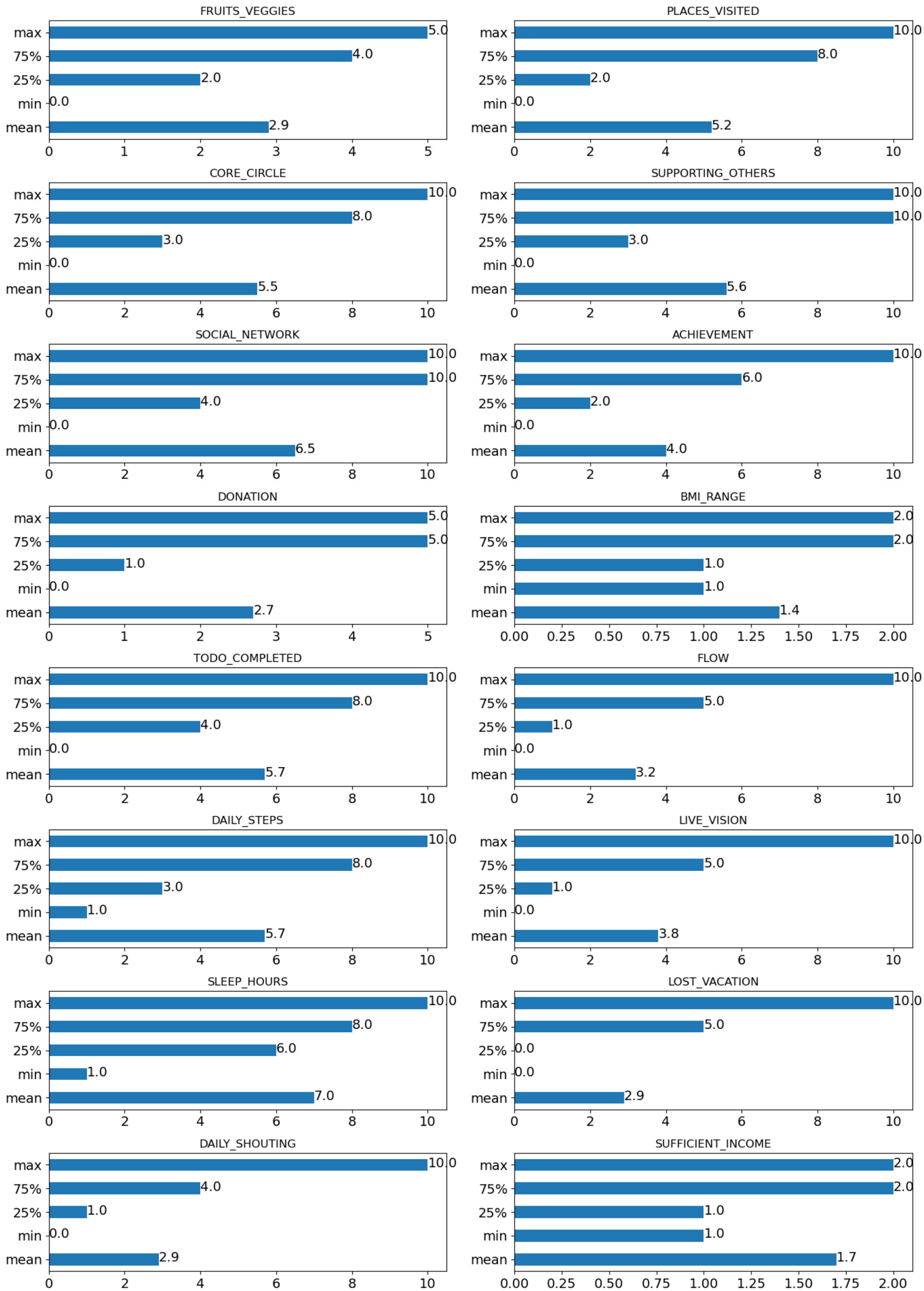
```
df['DAILY_STRESS'] = pd.to_numeric(df['DAILY_STRESS'],
errors='coerce')
df = df.dropna(subset=['DAILY_STRESS'])
df['DAILY_STRESS'] = df['DAILY_STRESS'].astype('int64')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 15971 entries, 0 to 15971
Data columns (total 24 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   Timestamp                15971 non-null  object
 1   FRUITS_VEGGIES           15971 non-null  int64
 2   DAILY_STRESS             15971 non-null  int64
 3   PLACES_VISITED           15971 non-null  int64
 4   CORE_CIRCLE              15971 non-null  int64
 5   SUPPORTING_OTHERS        15971 non-null  int64
 6   SOCIAL_NETWORK           15971 non-null  int64
 7   ACHIEVEMENT              15971 non-null  int64
 8   DONATION                 15971 non-null  int64
 9   BMI_RANGE                15971 non-null  int64
 10  TODO_COMPLETED           15971 non-null  int64
 11  FLOW                     15971 non-null  int64
 12  DAILY_STEPS              15971 non-null  int64
 13  LIVE_VISION              15971 non-null  int64
 14  SLEEP_HOURS              15971 non-null  int64
 15  LOST_VACATION            15971 non-null  int64
 16  DAILY_SHOUTING           15971 non-null  int64
 17  SUFFICIENT_INCOME        15971 non-null  int64
 18  PERSONAL_AWARDS          15971 non-null  int64
 19  TIME_FOR_PASSION         15971 non-null  int64
 20  WEEKLY_MEDITATION        15971 non-null  int64
 21  AGE                      15971 non-null  object
 22  GENDER                   15971 non-null  object
 23  WORK_LIFE_BALANCE_SCORE  15971 non-null  float64
dtypes: float64(1), int64(20), object(3)
memory usage: 3.0+ MB
```

```
df.describe()
```

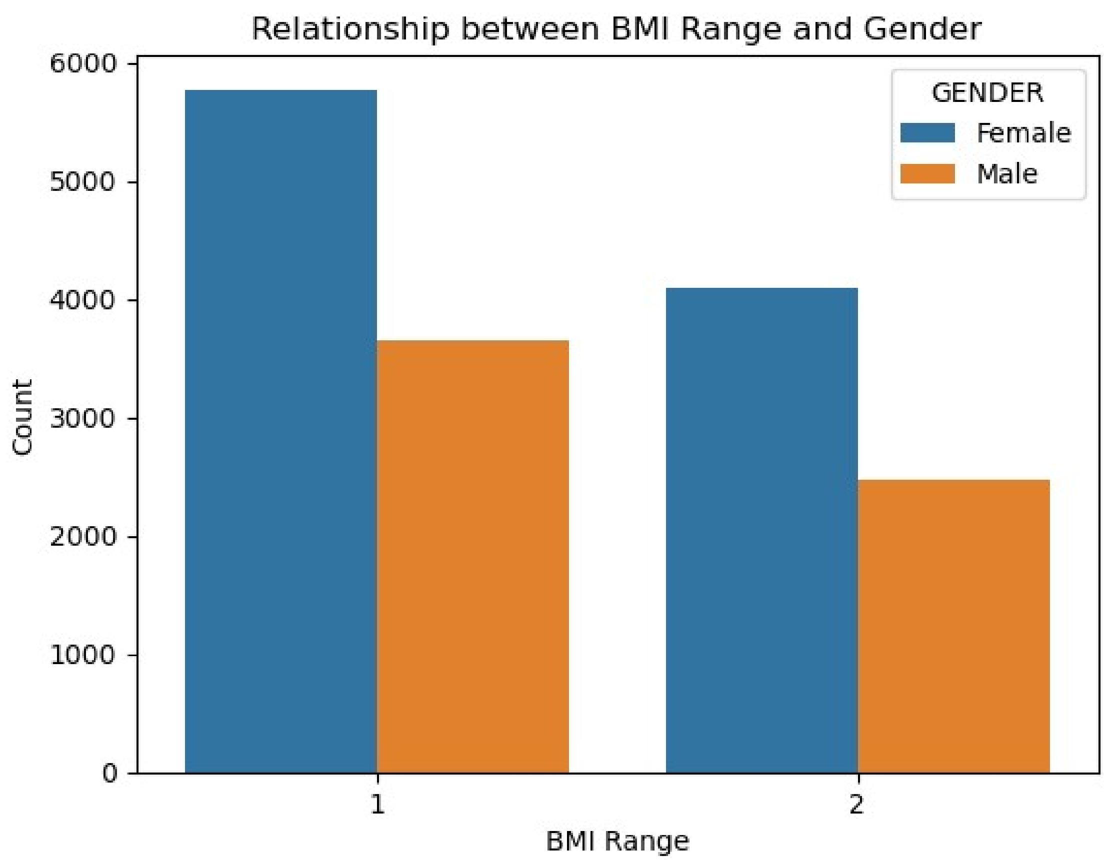
```
       FRUITS_VEGGIES  PLACES_VISITED   CORE_CIRCLE  SUPPORTING_OTHERS
\
count    15972.000000    15972.000000  15972.000000       15972.000000

mean         2.922677        5.232970      5.508077           5.616454

std          1.442694        3.311912      2.840334           3.242021

min          0.000000        0.000000      0.000000           0.000000

25%          2.000000        2.000000      3.000000           3.000000

50%          3.000000        5.000000      5.000000           5.000000

75%          4.000000        8.000000      8.000000          10.000000

max          5.000000       10.000000     10.000000          10.000000


       SOCIAL_NETWORK   ACHIEVEMENT      DONATION     BMI_RANGE  \
count    15972.000000  15972.000000  15972.000000  15972.000000
mean         6.474267      4.000751      2.715314      1.410656
std          3.086672      2.755837      1.851586      0.491968
min          0.000000      0.000000      0.000000      1.000000
25%          4.000000      2.000000      1.000000      1.000000
50%          6.000000      3.000000      3.000000      1.000000
75%         10.000000      6.000000      5.000000      2.000000
max         10.000000     10.000000      5.000000      2.000000

       TODO_COMPLETED          FLOW   DAILY_STEPS   LIVE_VISION
SLEEP_HOURS  \
count    15972.000000  15972.000000  15972.000000  15972.000000
15972.000000
mean         5.745993      3.194778      5.703606      3.752129
7.042888
std          2.624097      2.357518      2.891013      3.230987
1.199044
min          0.000000      0.000000      1.000000      0.000000
1.000000
25%          4.000000      1.000000      3.000000      1.000000
6.000000
50%          6.000000      3.000000      5.000000      3.000000
7.000000
75%          8.000000      5.000000      8.000000      5.000000
8.000000
max         10.000000     10.000000     10.000000     10.000000
10.000000

       LOST_VACATION  DAILY_SHOUTING  SUFFICIENT_INCOME
PERSONAL_AWARDS  \
count   15972.000000    15972.000000       15972.000000
```

```
15972.000000
mean          2.898886          2.930879              1.728963
5.711558
std           3.692180          2.676301              0.444509
3.089630
min           0.000000          0.000000              1.000000
0.000000
25%           0.000000          1.000000              1.000000
3.000000
50%           0.000000          2.000000              2.000000
5.000000
75%           5.000000          4.000000              2.000000
9.000000
max          10.000000         10.000000              2.000000
10.000000

       TIME_FOR_PASSION  WEEKLY_MEDITATION  WORK_LIFE_BALANCE_SCORE
count      15972.000000       15972.000000             15972.000000
mean           3.326572           6.233346               666.751503
std            2.729293           3.016571                45.019868
min            0.000000           0.000000               480.000000
25%            1.000000           4.000000               636.000000
50%            3.000000           7.000000               667.700000
75%            5.000000          10.000000               698.500000
max           10.000000          10.000000               820.200000

def descriptive(df):
    desc=df.describe().round(1).drop({'count', 'std', '50%'}, axis=0)
    i=-0.1
    j=0
    Row = int(round(len(desc.columns.tolist())/2+0.1))
    f,ax = plt.subplots(Row,2, figsize=(28,18))
    for name in desc.columns.tolist():
        desc[name].plot(kind='barh', figsize=(14,24), title=name, 
ax=ax[round(i), j], fontsize=14)
        for k, v in enumerate(desc[name].tolist()):
            ax[round(i), j].text(v, k-0.1, str(v), color='black', size
= 14)
        i +=0.5
        if j==0: j=1
        else: j=0
    f.tight_layout()
descriptive(df)
```

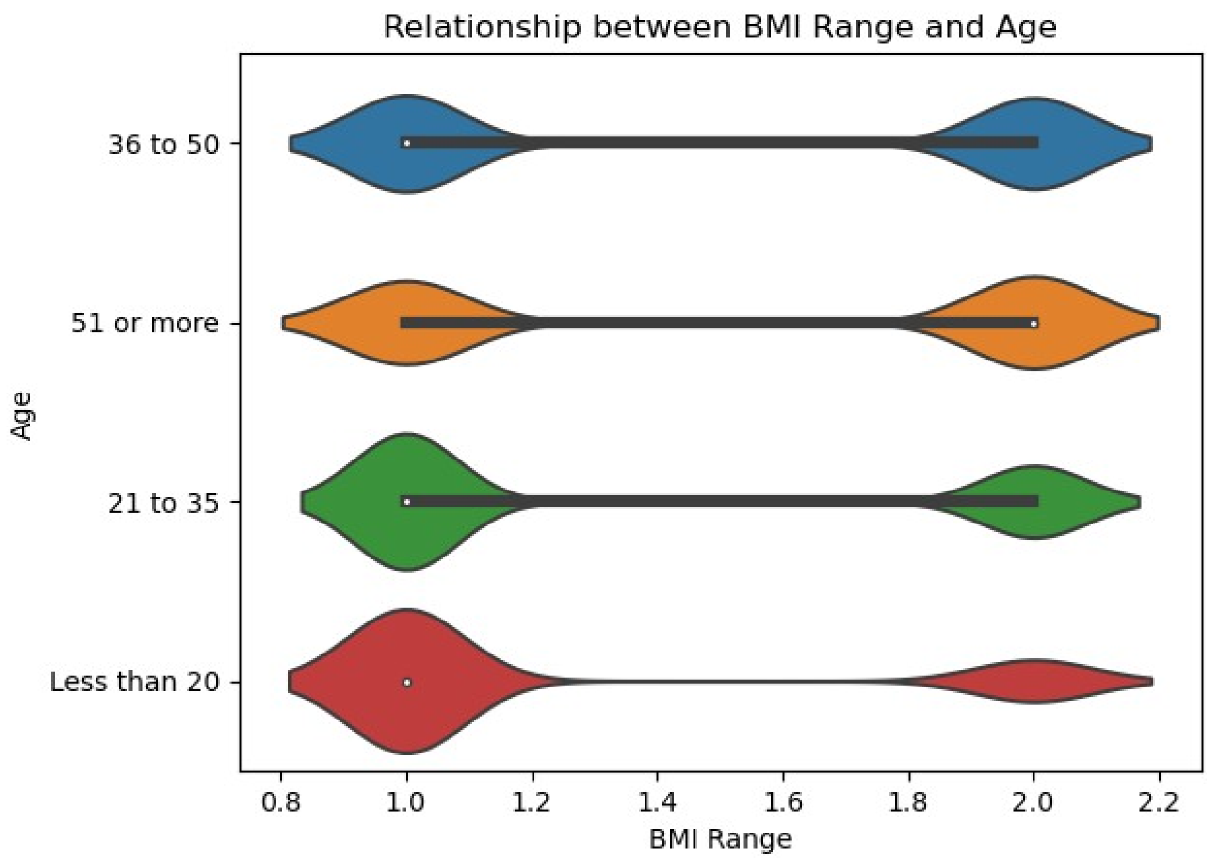

FRUITS_VEGGIES
max 5.0
75% 4.0
25% 2.0
min 0.0
mean 2.9
PLACES_VISITED
max 10.0
75% 8.0
25% 2.0
min 0.0
mean 5.2
CORE_CIRCLE
max 10.0
75% 8.0
25% 3.0
min 0.0
mean 5.5
SUPPORTING_OTHERS
max 10.0
75% 10.0
25% 3.0
min 0.0
mean 5.6
SOCIAL_NETWORK
max 10.0
75% 10.0
25% 4.0
min 0.0
mean 6.5
ACHIEVEMENT
max 10.0
75% 6.0
25% 2.0
min 0.0
mean 4.0
DONATION
max 5.0
75% 5.0
25% 1.0
min 0.0
mean 2.7
BMI_RANGE
max 2.0
75% 2.0
25% 1.0
min 1.0
mean 1.4
TODO_COMPLETED
max 10.0
75% 8.0
25% 4.0
min 0.0
mean 5.7
FLOW
max 10.0
75% 5.0
25% 1.0
min 0.0
mean 3.2
DAILY_STEPS
max 10.0
75% 8.0
25% 3.0
min 1.0
mean 5.7
LIVE_VISION
max 10.0
75% 5.0
25% 1.0
min 0.0
mean 3.8
SLEEP_HOURS
max 10.0
75% 8.0
25% 6.0
min 1.0
mean 7.0
LOST_VACATION
max 10.0
75% 5.0
25% 0.0
min 0.0
mean 2.9
DAILY_SHOUTING
max 10.0
75% 4.0
25% 1.0
min 0.0
mean 2.9
SUFFICIENT_INCOME
max 2.0
75% 2.0
25% 1.0
min 1.0
mean 1.7

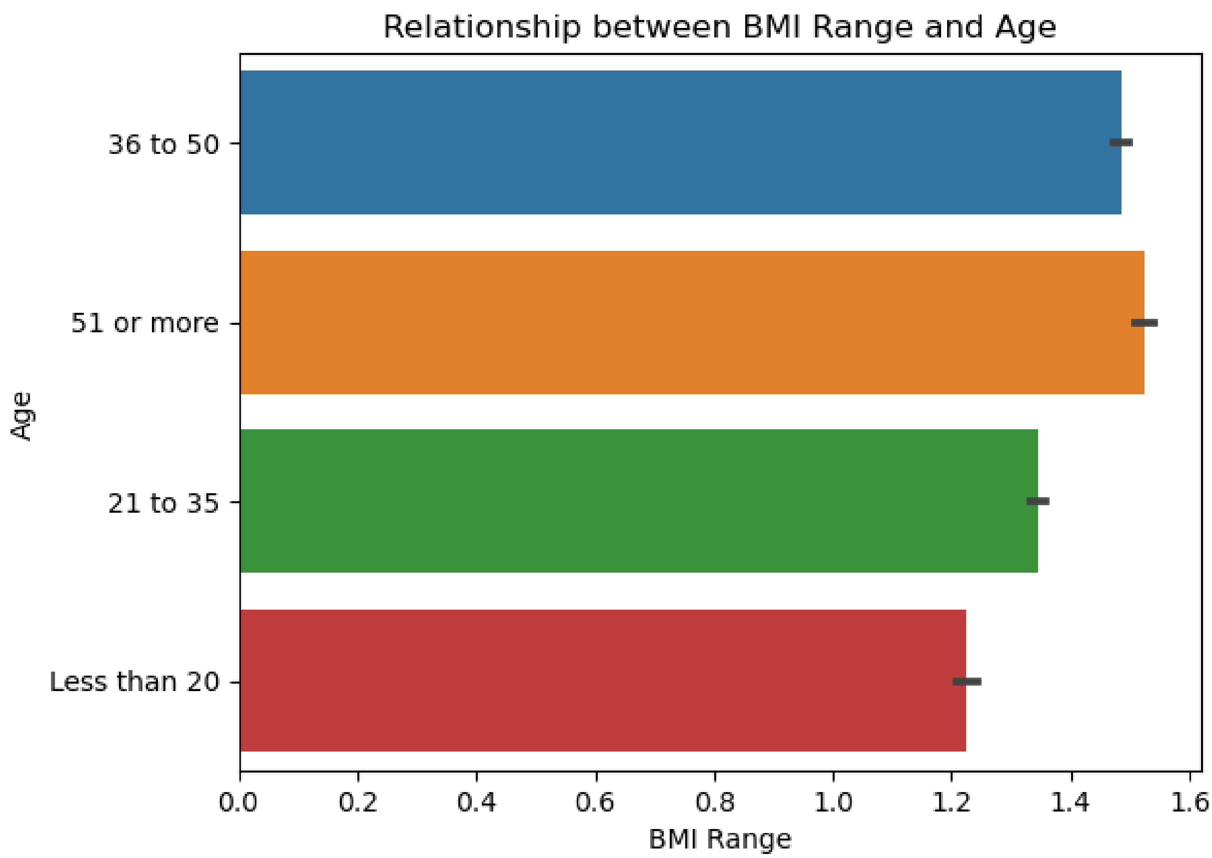
Analysis of factors correlating to Physical Health

```
sns.countplot(x='BMI_RANGE', hue='GENDER', data=df)
plt.title('Relationship between BMI Range and Gender')
plt.xlabel('BMI Range')
plt.ylabel('Count')
plt.show()
```

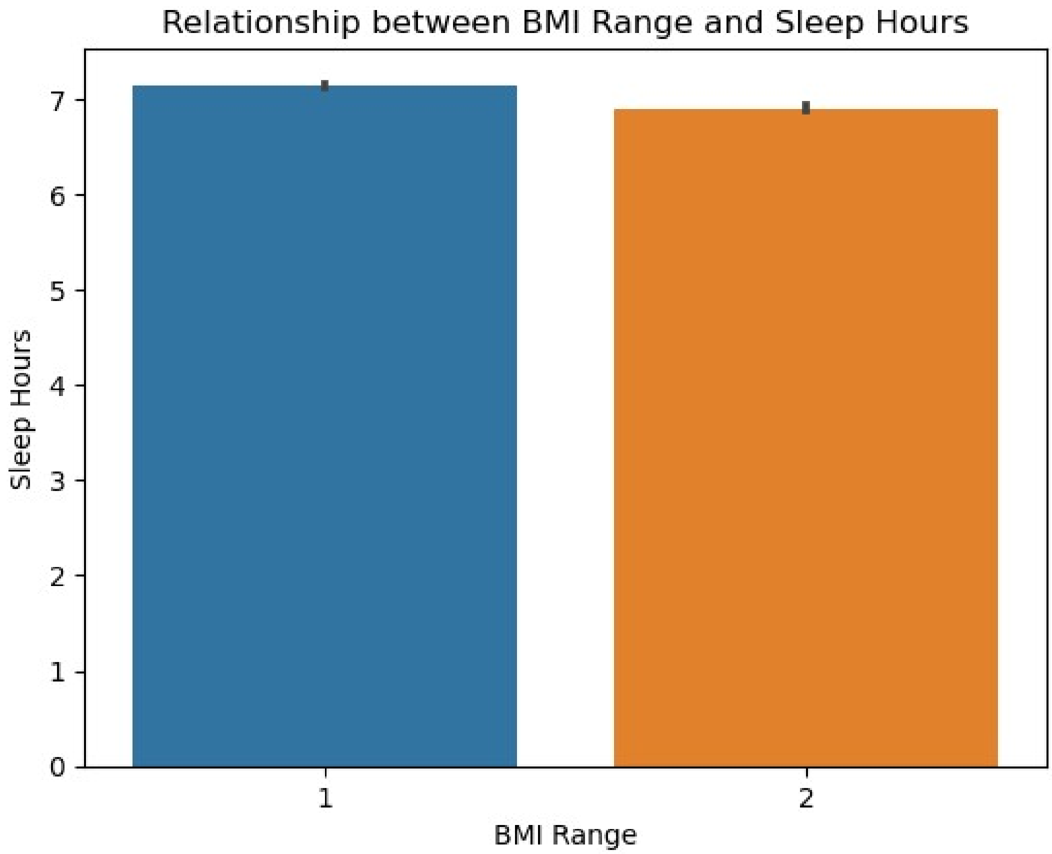



```
sns.violinplot(x='BMI_RANGE', y='AGE', data=df)
plt.title('Relationship between BMI Range and Age')
plt.xlabel('BMI Range')
plt.ylabel('Age')
plt.show()

sns.barplot(x='BMI_RANGE', y='AGE', data=df)
plt.title('Relationship between BMI Range and Age')
plt.xlabel('BMI Range')
plt.ylabel('Age')
plt.show()
```

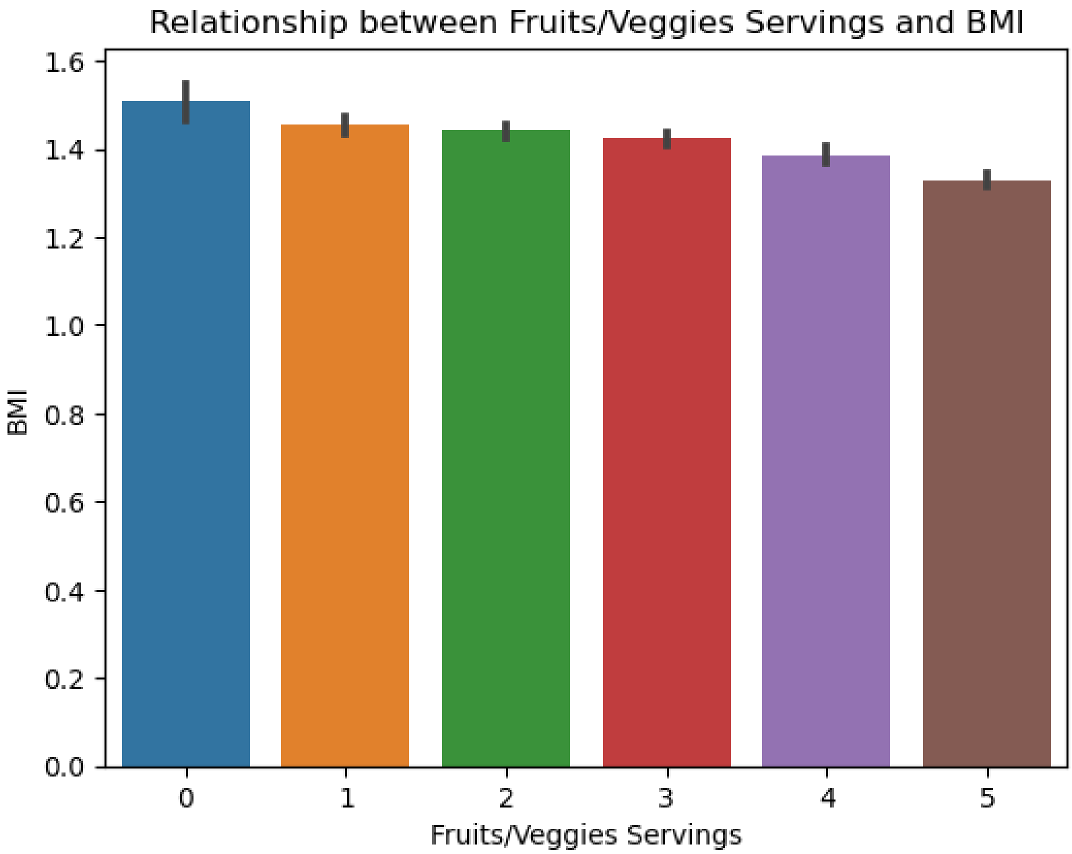

Relationship between BMI Range and Age
36 to 50
51 or more
21 to 35
Less than 20
Age
0.8
1.0
1.2
1.4
1.6
1.8
2.0
2.2
BMI Range



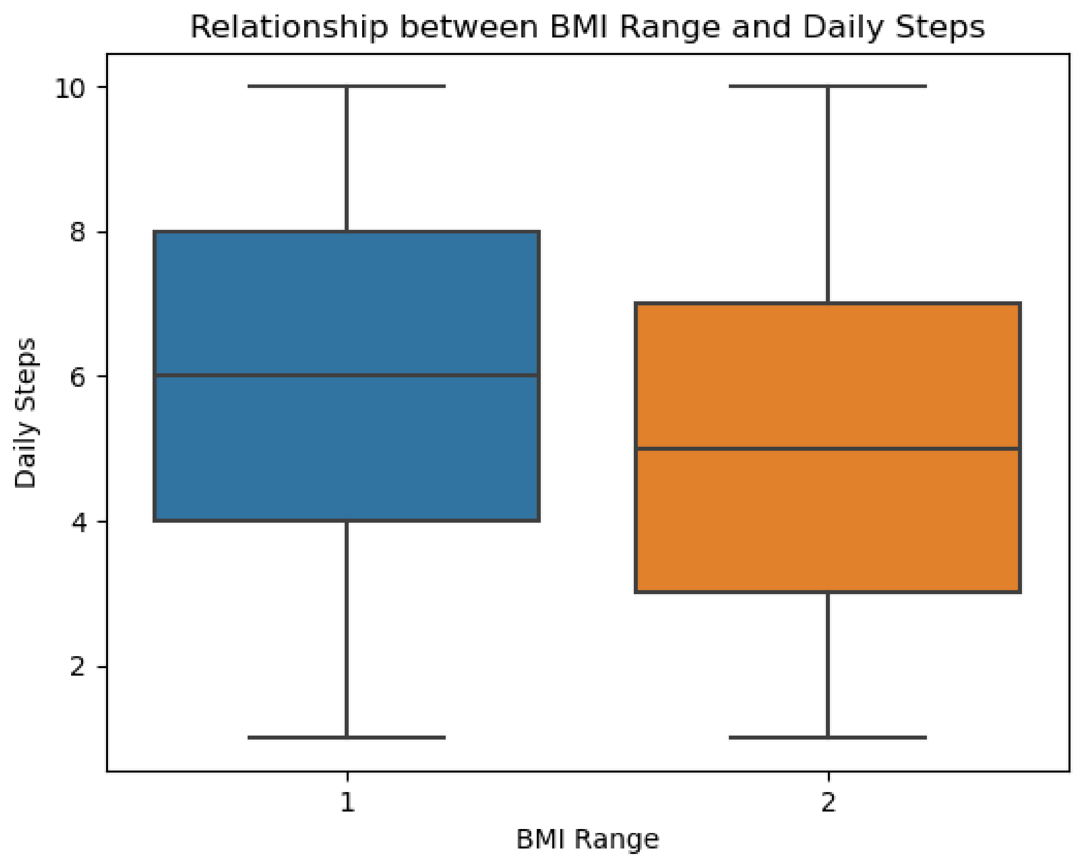

```
sns.barplot(x='BMI_RANGE', y='SLEEP_HOURS', data=df)
plt.title('Relationship between BMI Range and Sleep Hours')
plt.xlabel('BMI Range')
plt.ylabel('Sleep Hours')
plt.show()
```

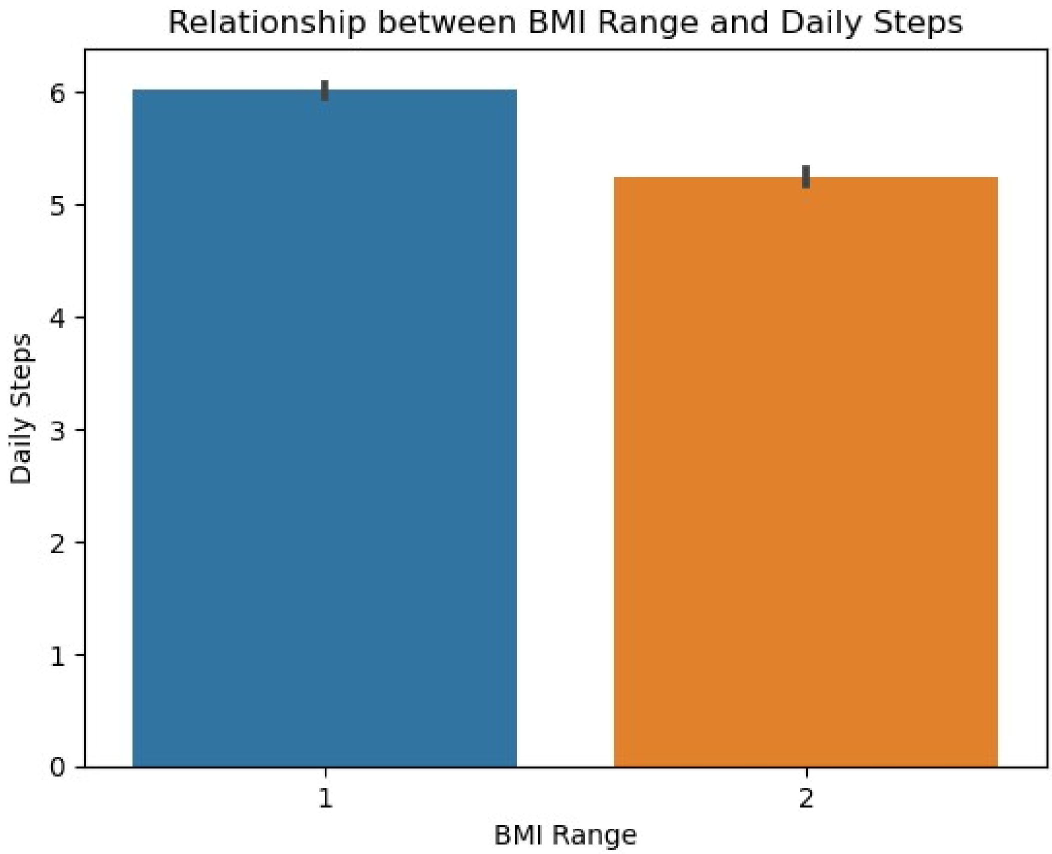



```
sns.barplot(x='FRUITS_VEGGIES', y='BMI_RANGE', data=df)
plt.title('Relationship between Fruits/Veggies Servings and BMI')
plt.xlabel('Fruits/Veggies Servings')
plt.ylabel('BMI')
plt.show()
```

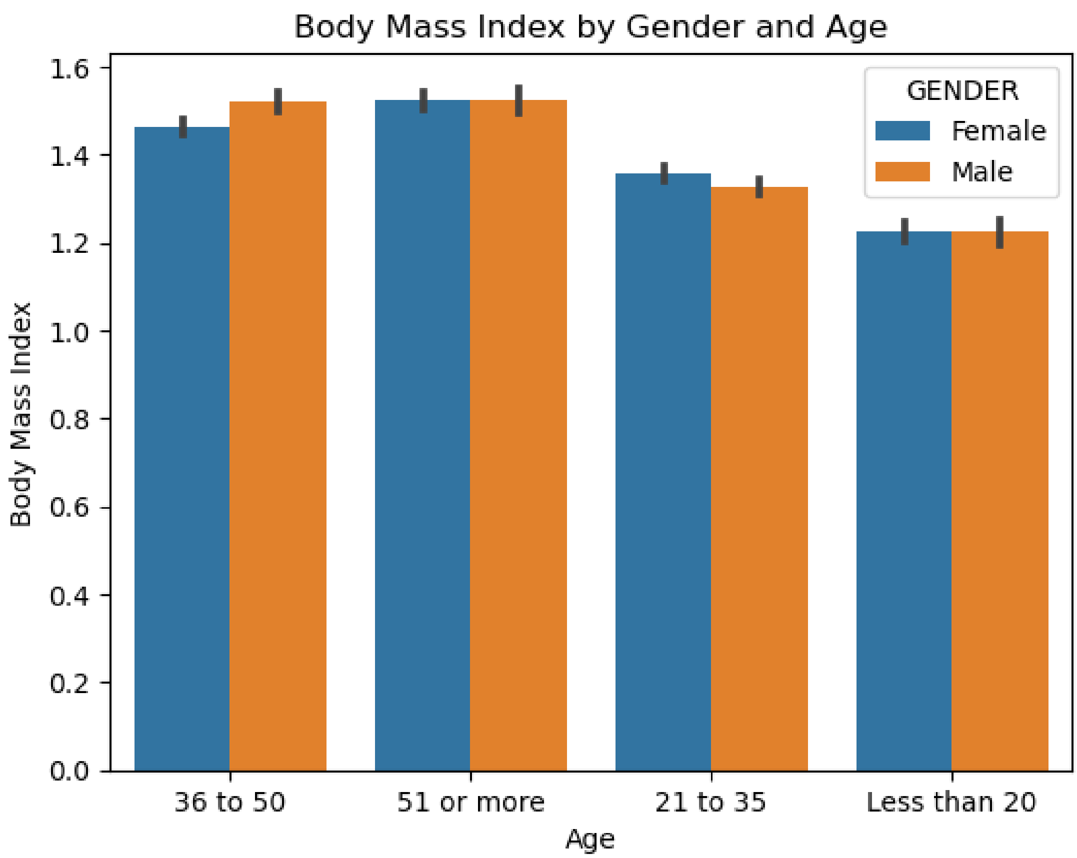



```
sns.boxplot(x='BMI_RANGE', y='DAILY_STEPS', data=df)
plt.title('Relationship between BMI Range and Daily Steps')
plt.xlabel('BMI Range')
plt.ylabel('Daily Steps')
plt.show()

sns.barplot(x='BMI_RANGE', y='DAILY_STEPS', data=df)
plt.title('Relationship between BMI Range and Daily Steps')
plt.xlabel('BMI Range')
plt.ylabel('Daily Steps')
plt.show()
```


Relationship between BMI Range and Daily Steps
10
8
6
4
2
Daily Steps
1
2
BMI Range




```
sns.barplot(x='AGE', y='BMI_RANGE', hue='GENDER', data=df)
plt.title('Body Mass Index by Gender and Age')
plt.xlabel('Age')
plt.ylabel('Body Mass Index')
plt.show()
```



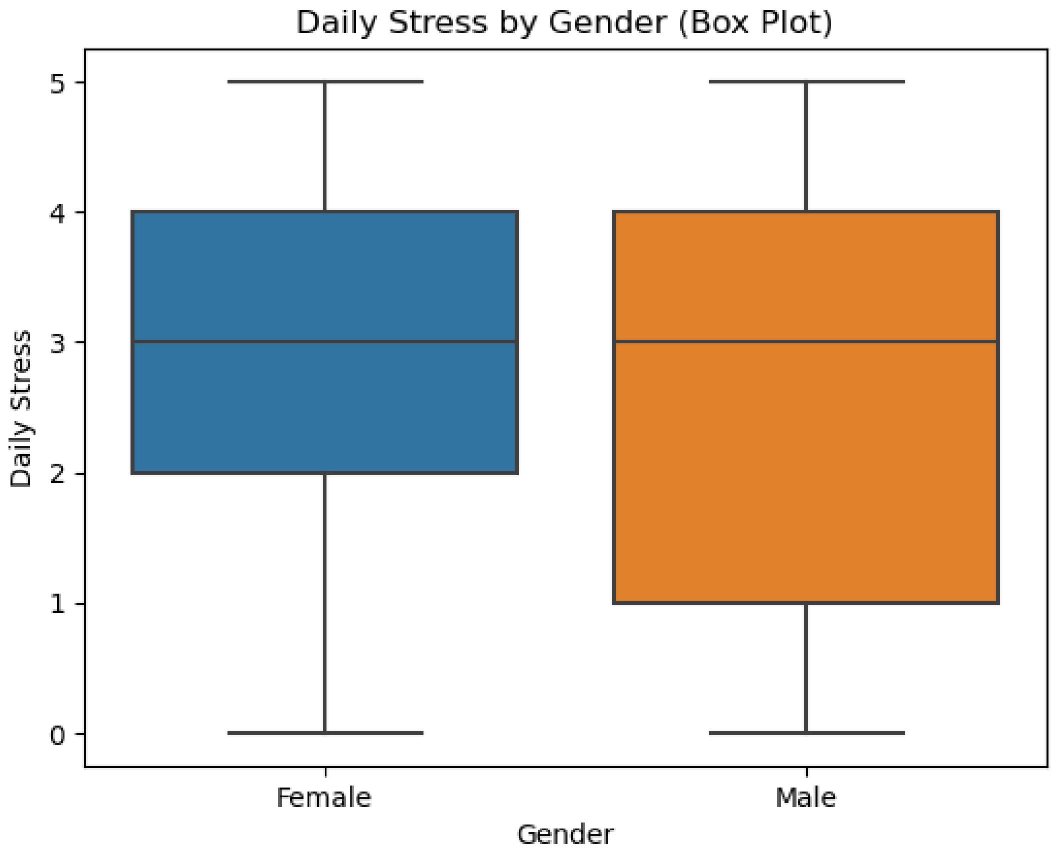
Conclusion: BMI is strongly correlated to daily steps and servings of fruits & vegetables so we can say that physical activity and an healthy diet contribute to a lower BMI.

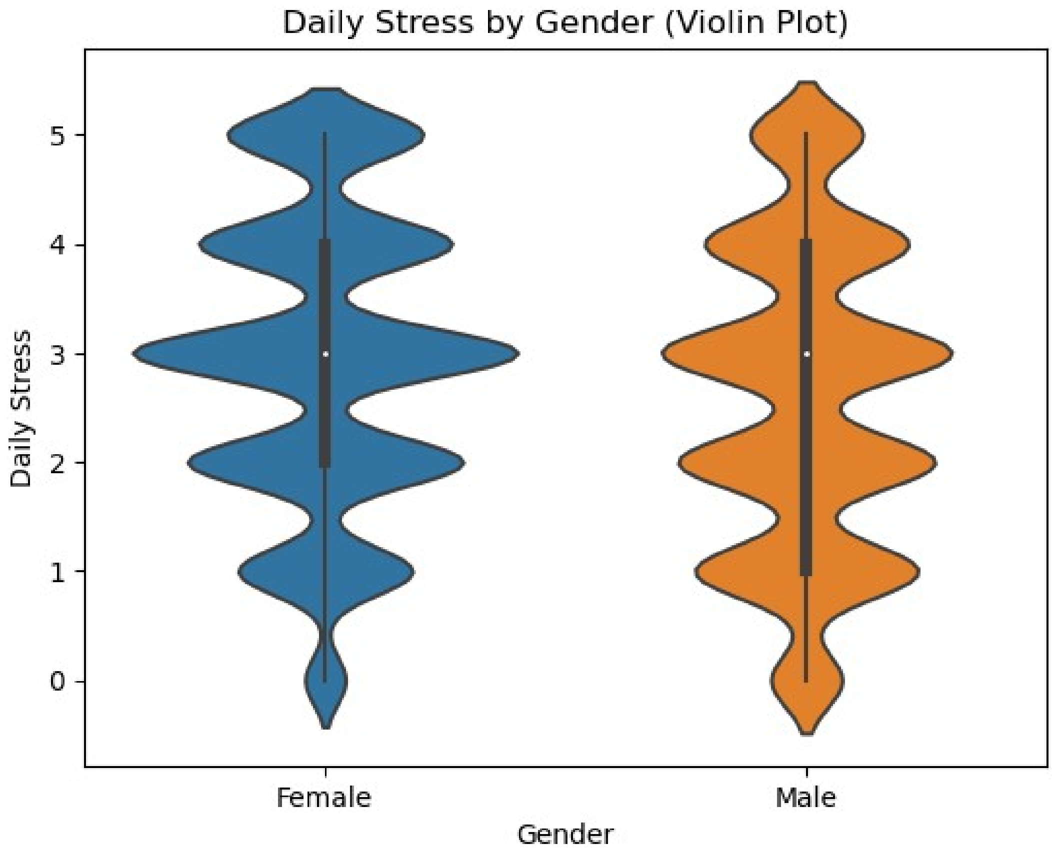
The BMI for men and women average to very close values for age groups "less than 20" and "51 or more". But stronger differences are found for the following age ranges: 21 to 35: women's BMI are higher 36 to 50: men's BMI are higher

Analysis of factors correlating to Mental Health

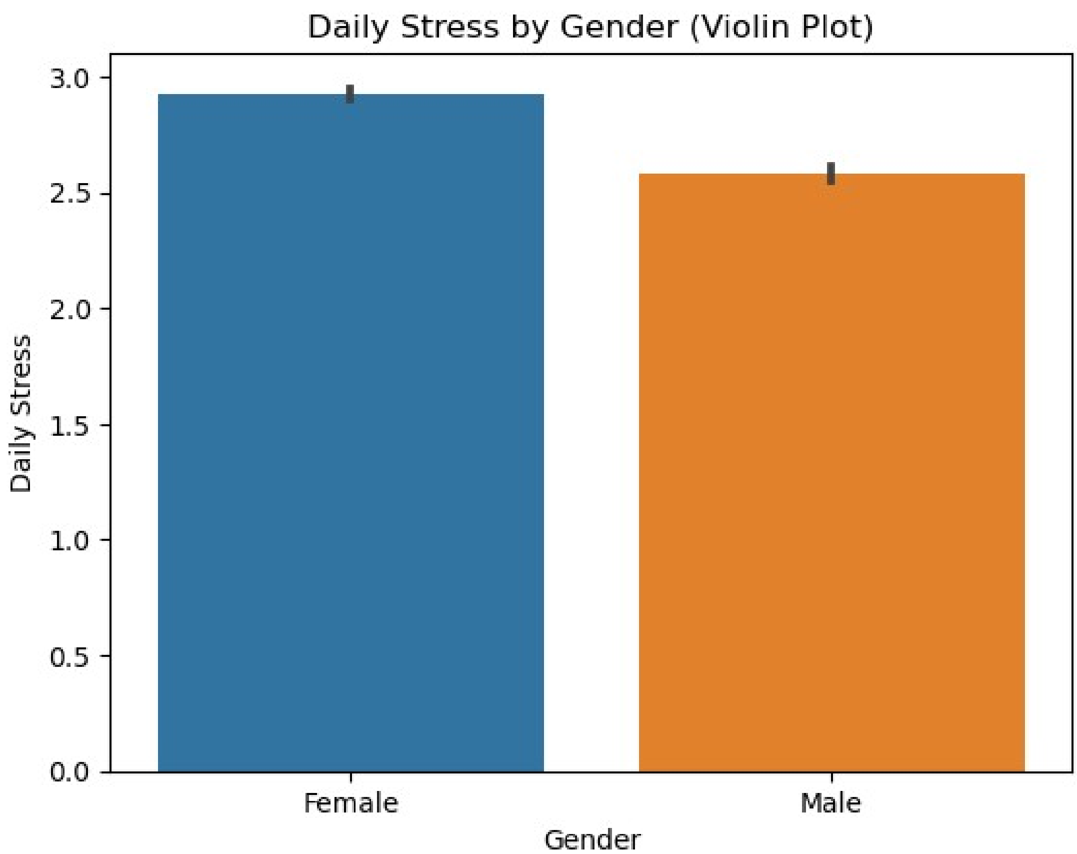
```
sns.boxplot(x='GENDER', y='DAILY_STRESS', data=df)
plt.title('Daily Stress by Gender (Box Plot)')
plt.xlabel('Gender')
plt.ylabel('Daily Stress')
plt.show()

sns.violinplot(x='GENDER', y='DAILY_STRESS', data=df)
plt.title('Daily Stress by Gender (Violin Plot)')
plt.xlabel('Gender')
plt.ylabel('Daily Stress')
plt.show()

sns.barplot(x='GENDER', y='DAILY_STRESS', data=df)
plt.title('Daily Stress by Gender (Violin Plot)')
plt.xlabel('Gender')
```

```
plt.ylabel('Daily Stress')
plt.show()
```

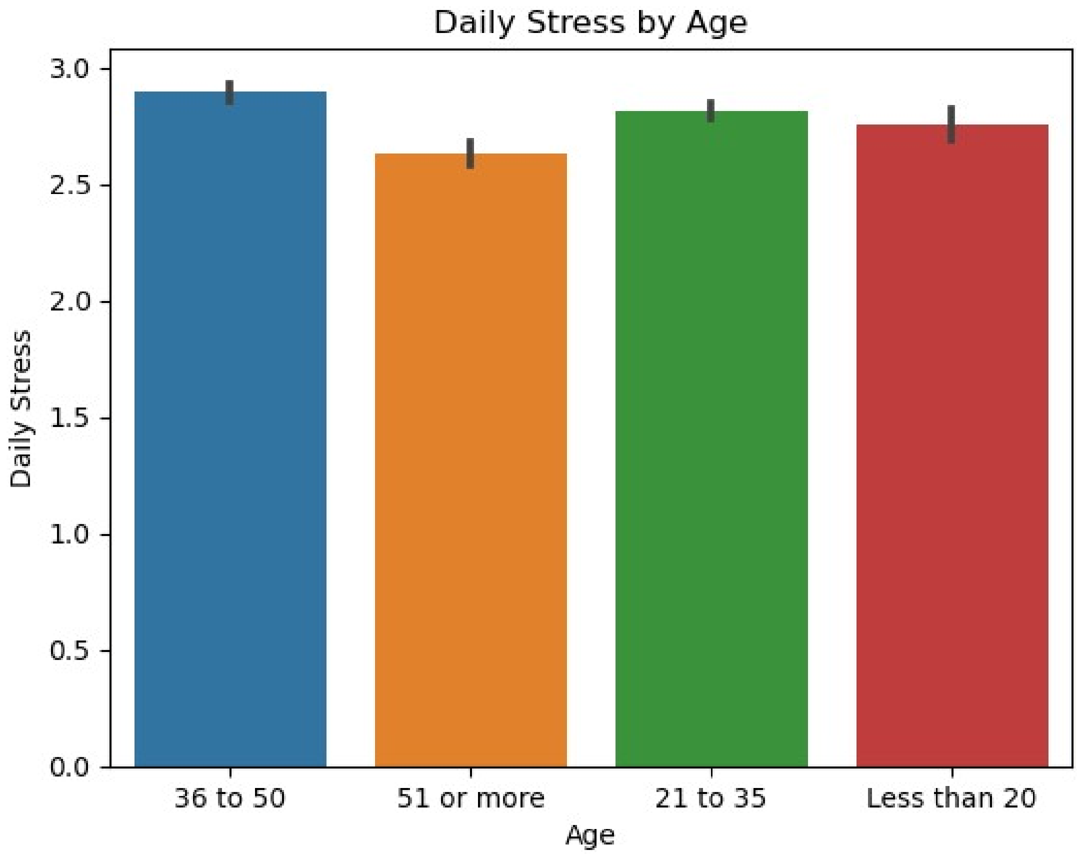




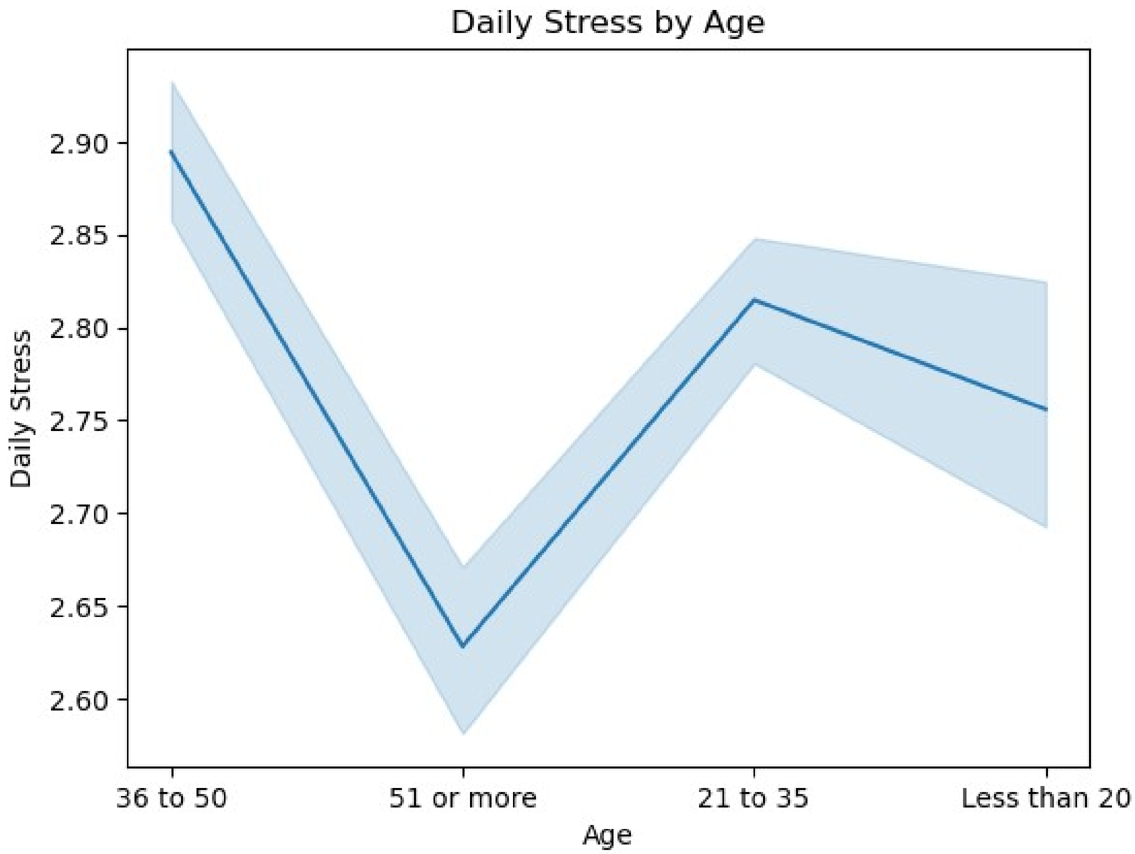
Daily Stress by Gender (Violin Plot)
Daily Stress
5
4
3
2
1
0
Female
Male
Gender



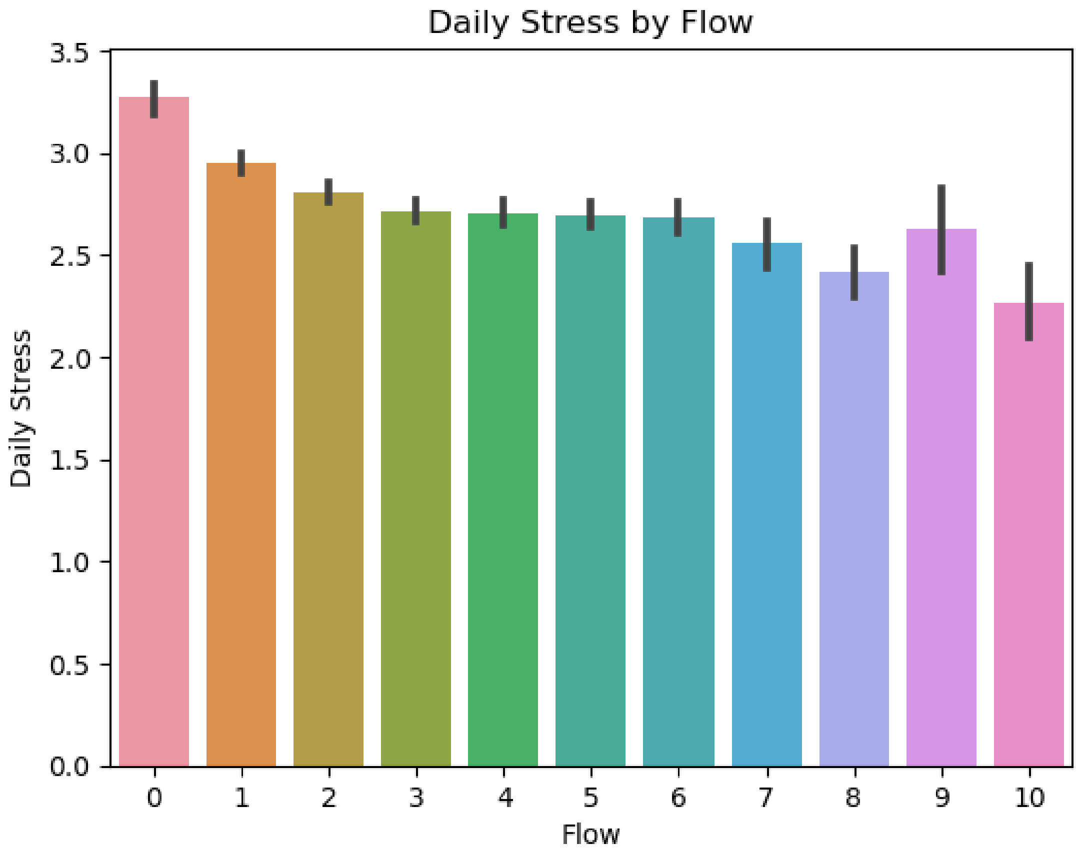

```
sns.barplot(x='AGE', y='DAILY_STRESS', data=df)
plt.title('Daily Stress by Age')
plt.xlabel('Age')
plt.ylabel('Daily Stress')
plt.show()

sns.lineplot(x='AGE', y='DAILY_STRESS', data=df)
plt.title('Daily Stress by Age')
plt.xlabel('Age')
plt.ylabel('Daily Stress')
plt.show()
```



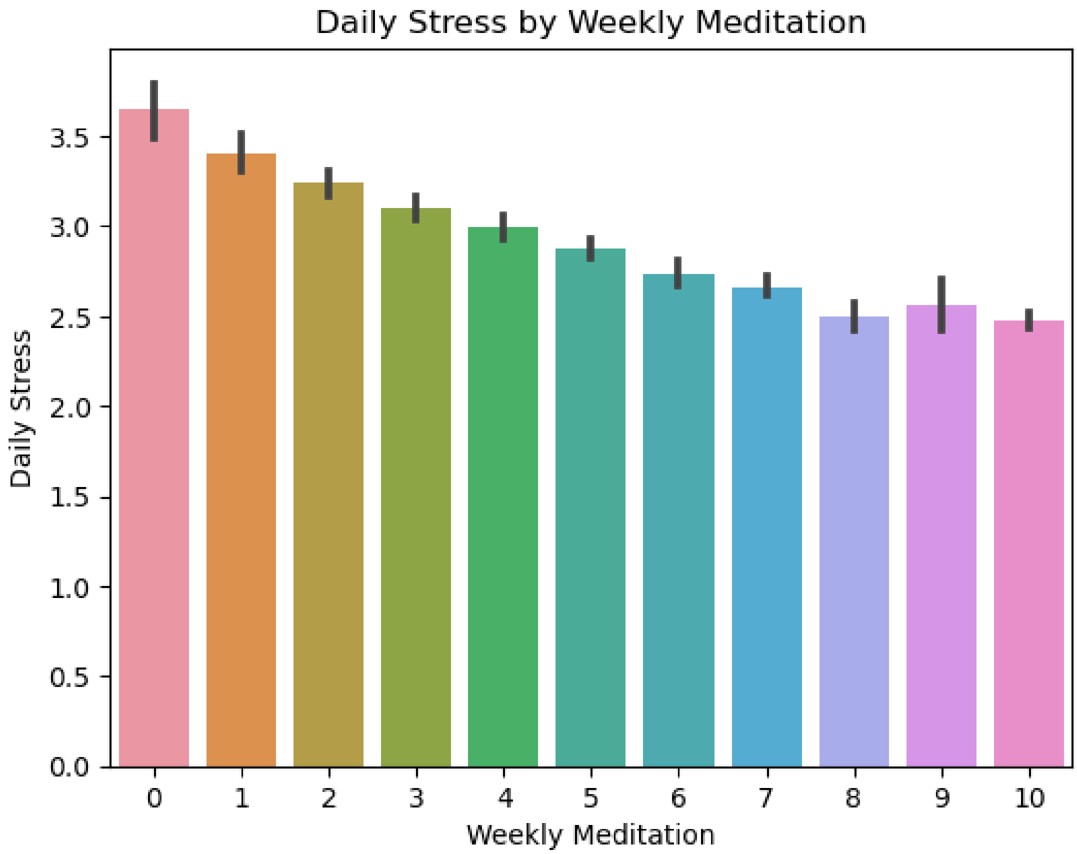

```
c:\Users\siddh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\siddh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



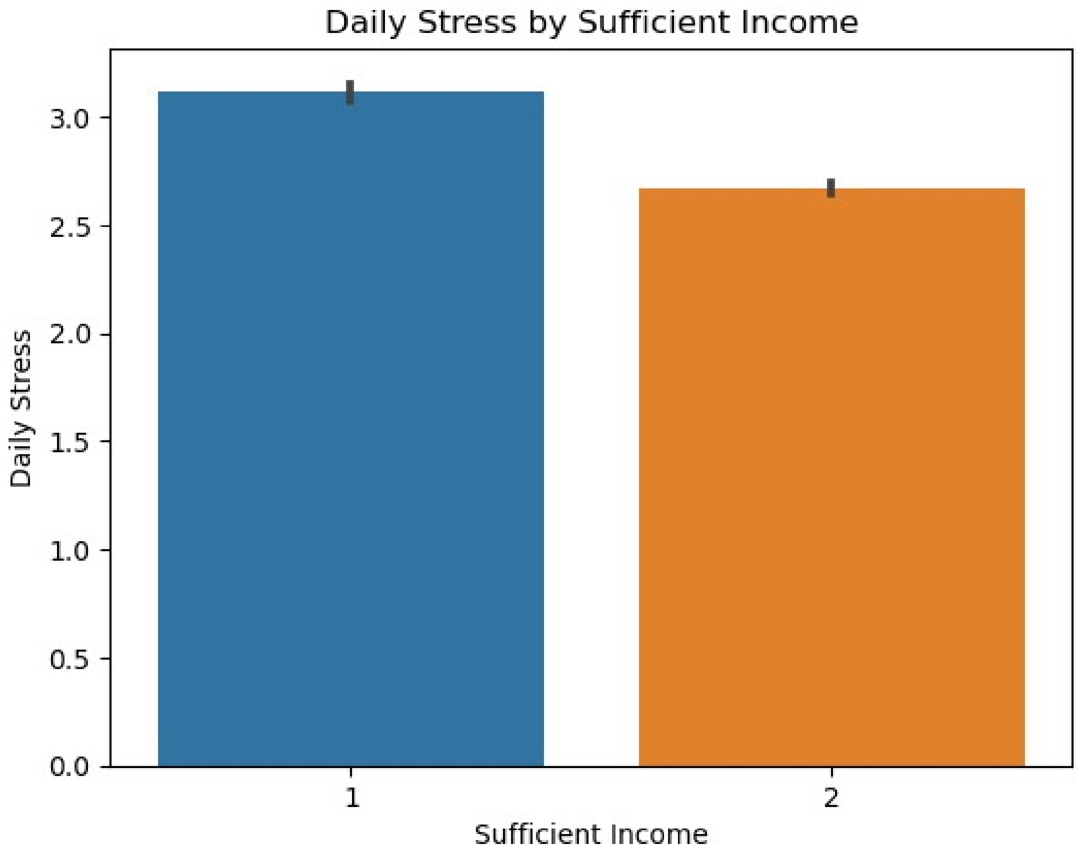

```
sns.barplot(x='FLOW', y='DAILY_STRESS', data=df)
plt.title('Daily Stress by Flow')
plt.xlabel('Flow')
plt.ylabel('Daily Stress')
plt.show()
```




```
sns.barplot(x='WEEKLY_MEDITATION', y='DAILY_STRESS', data=df)
plt.title('Daily Stress by Weekly Meditation')
plt.xlabel('Weekly Meditation')
plt.ylabel('Daily Stress')
plt.show()
```

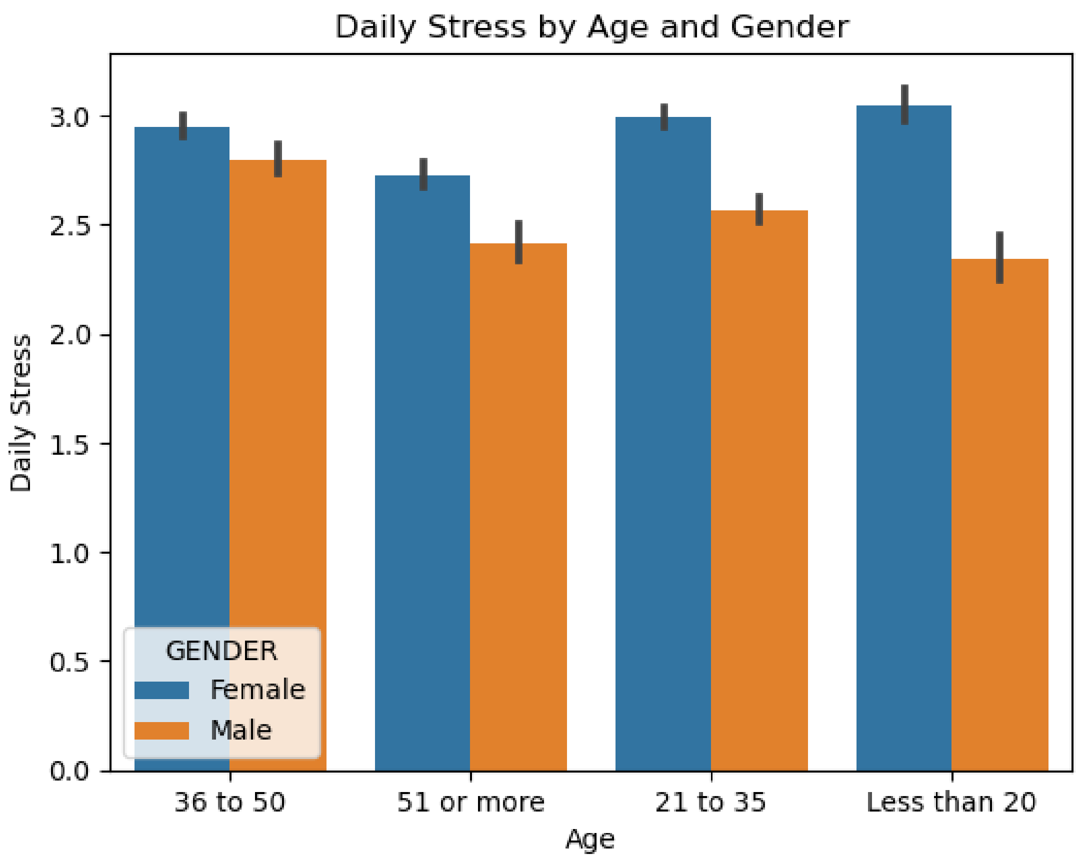



```
sns.barplot(x='SUFFICIENT_INCOME', y='DAILY_STRESS', data=df)
plt.title('Daily Stress by Sufficient Income')
plt.xlabel('Sufficient Income')
plt.ylabel('Daily Stress')
plt.show()
```



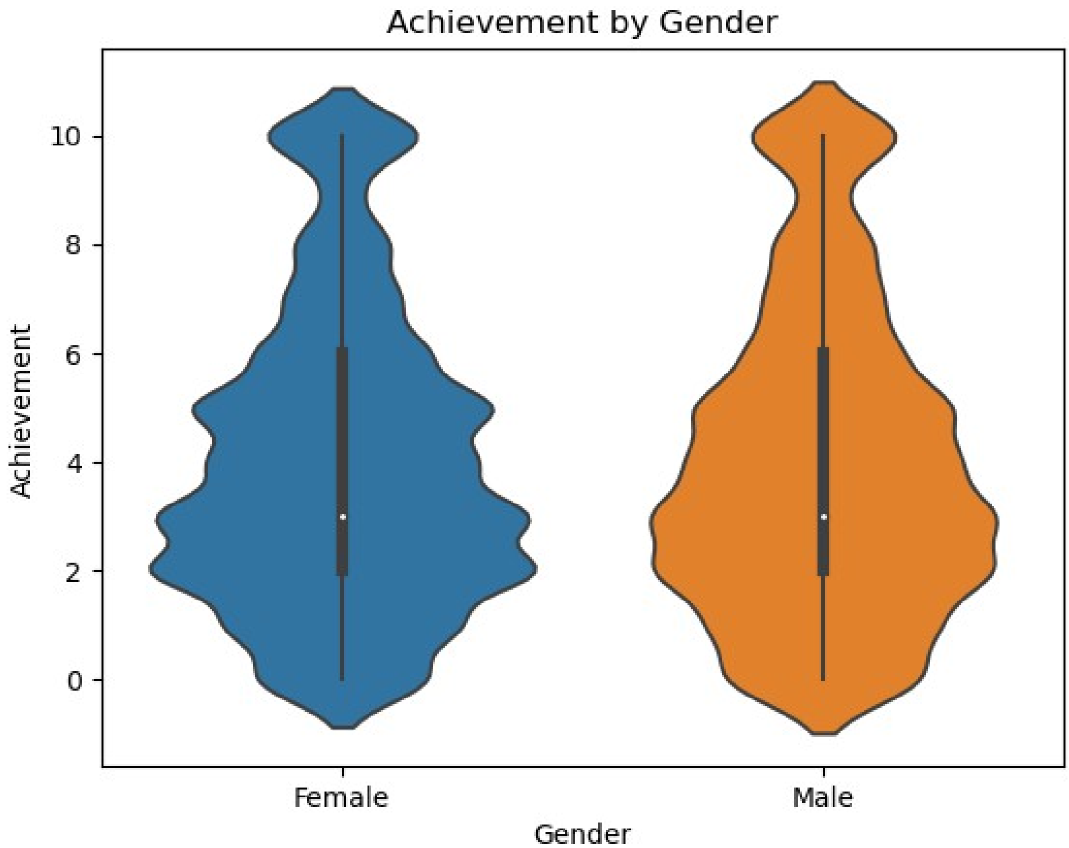

```
sns.violinplot(x='AGE', y='DAILY_STRESS', hue='GENDER', data=df,
split=True)
plt.title('Daily Stress by Age and Gender')
plt.xlabel('Age')
plt.ylabel('Daily Stress')
plt.show()

sns.barplot(x='AGE', y='DAILY_STRESS', hue='GENDER', data=df)
plt.title('Daily Stress by Age and Gender')
plt.xlabel('Age')
plt.ylabel('Daily Stress')
plt.show()
```

Daily Stress by Age and Gender
Daily Stress
6
5
4
3
2
1
0
−1
GENDER
Female
Male
36 to 50
51 or more
21 to 35
Less than 20
Age

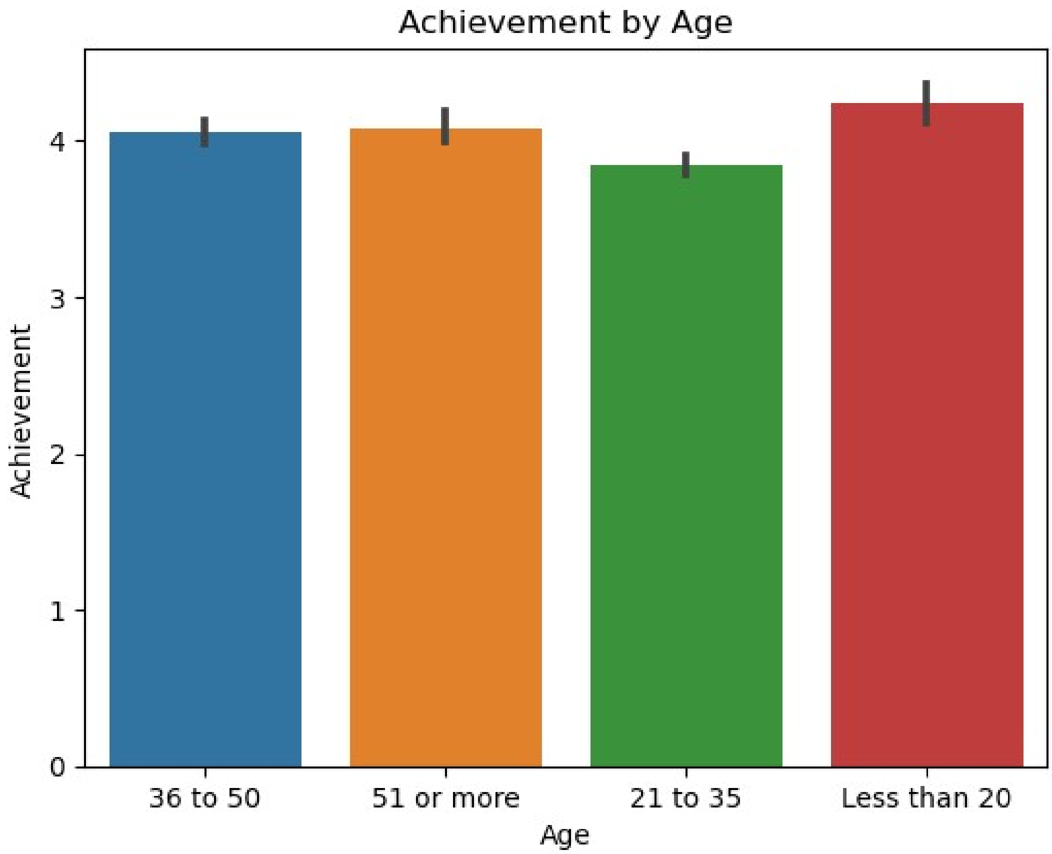


Conclusion: How ability to "flow" during the day, daily meditation, and an income sufficient to cover basic needs, all contribute to lower levels of stress. The overall stress level for women peaks in their younger years, and, while slowly going down remains higher than the male counterparts in all age groups.

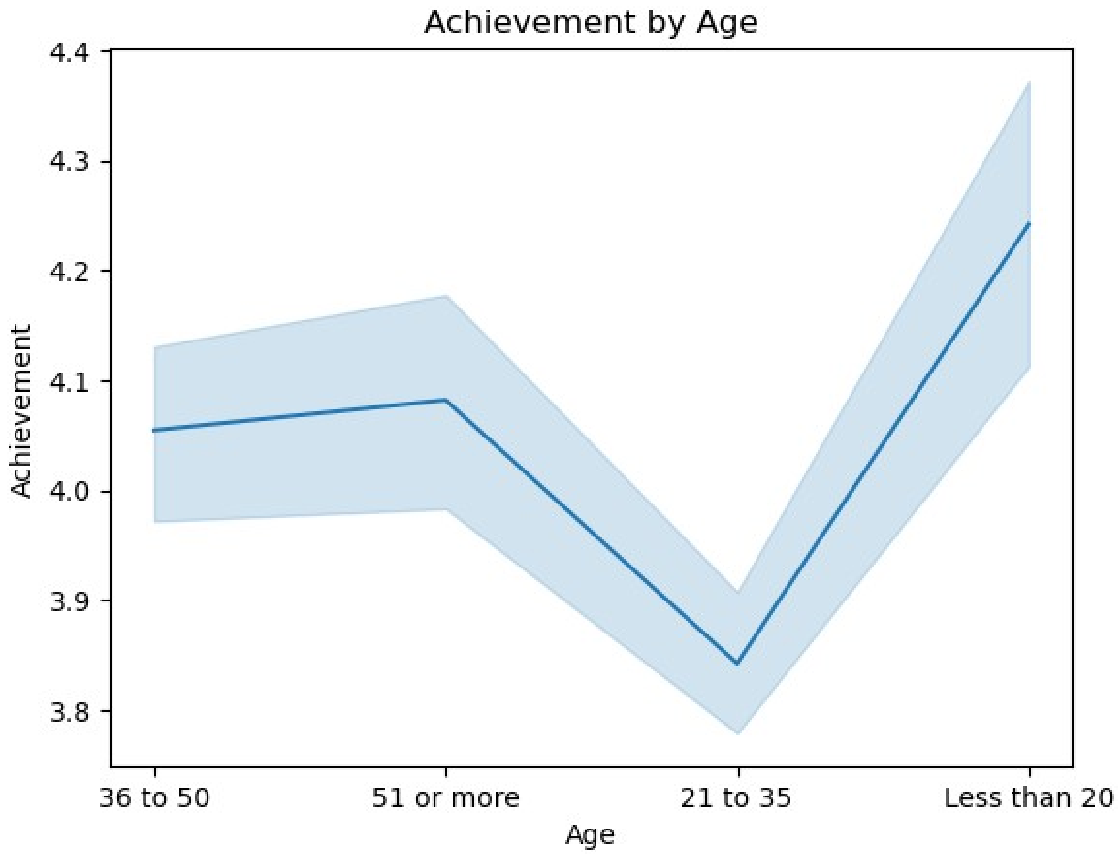
```
Analysis of factors correlating to Expertise

sns.violinplot(x='GENDER', y='ACHIEVEMENT', data=df)
plt.title('Achievement by Gender')
plt.xlabel('Gender')
plt.ylabel('Achievement')
plt.show()
```



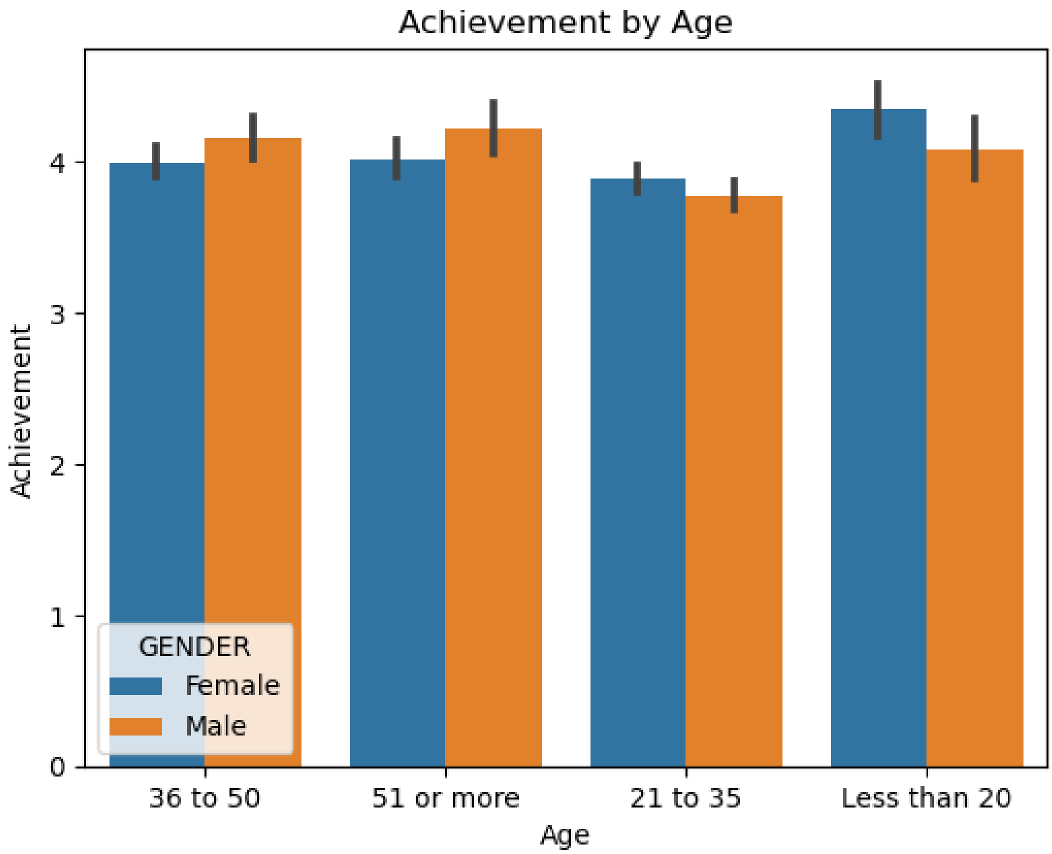

```
sns.barplot(x='AGE', y='ACHIEVEMENT', data=df)
plt.title('Achievement by Age')
plt.xlabel('Age')
plt.ylabel('Achievement')
plt.show()

sns.lineplot(x='AGE', y='ACHIEVEMENT', data=df)
plt.title('Achievement by Age')
plt.xlabel('Age')
plt.ylabel('Achievement')
plt.show()
```



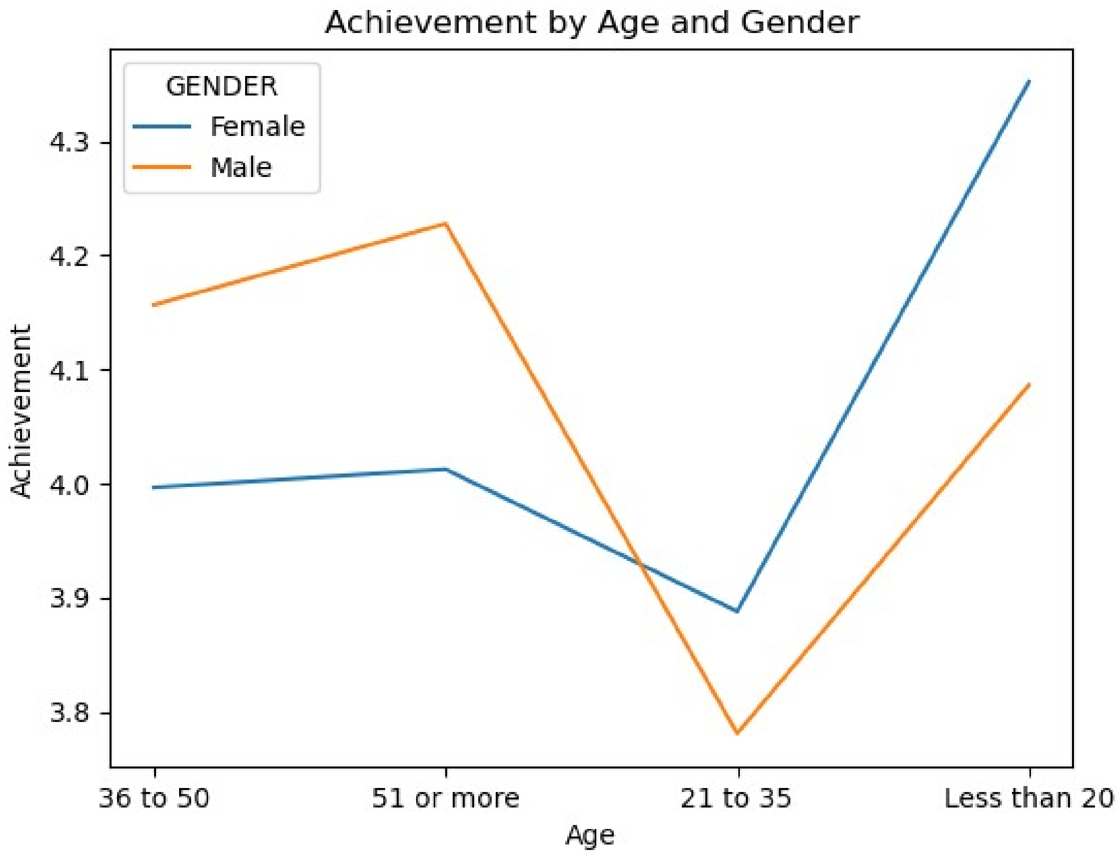

```
c:\Users\siddh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\siddh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



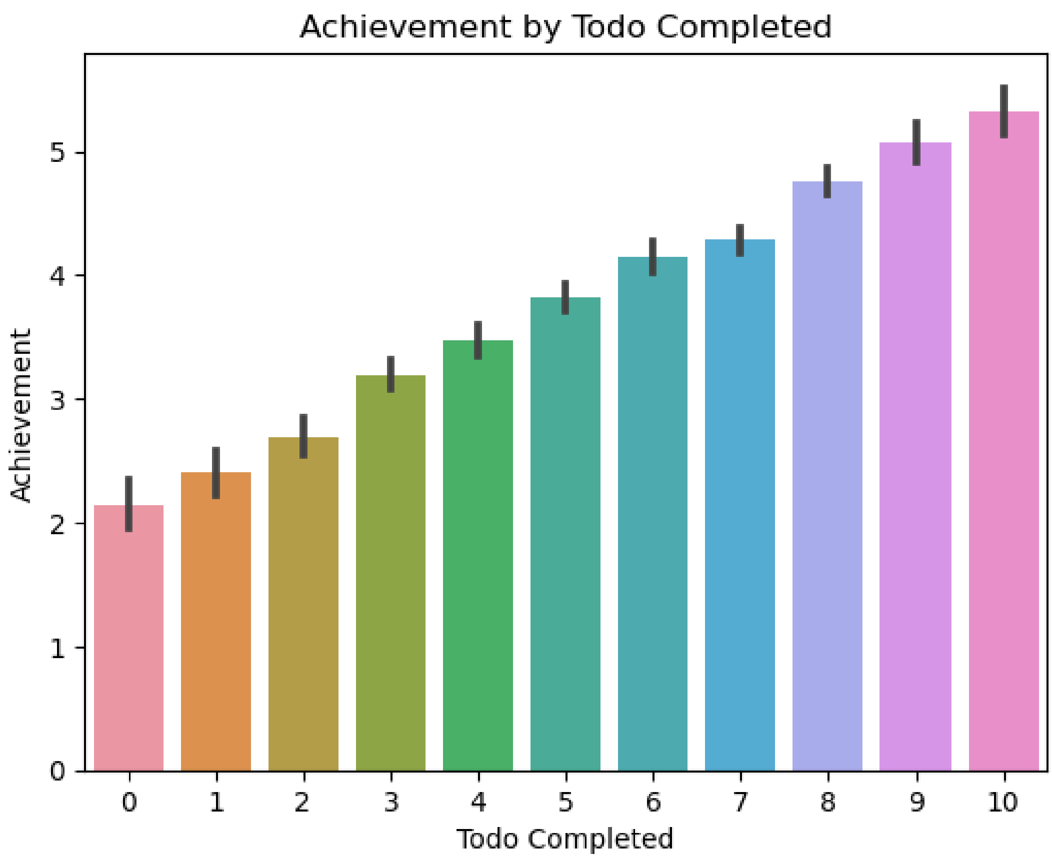

```
sns.barplot(x='AGE', y='ACHIEVEMENT', hue='GENDER' , data=df)
plt.title('Achievement by Age')
plt.xlabel('Age')
plt.ylabel('Achievement')
plt.show()

sns.lineplot(x='AGE', y='ACHIEVEMENT', hue='GENDER', data=df, ci=None)
# ci=None removes confidence intervals
plt.title('Achievement by Age and Gender')
plt.xlabel('Age')
plt.ylabel('Achievement')
plt.show()
```



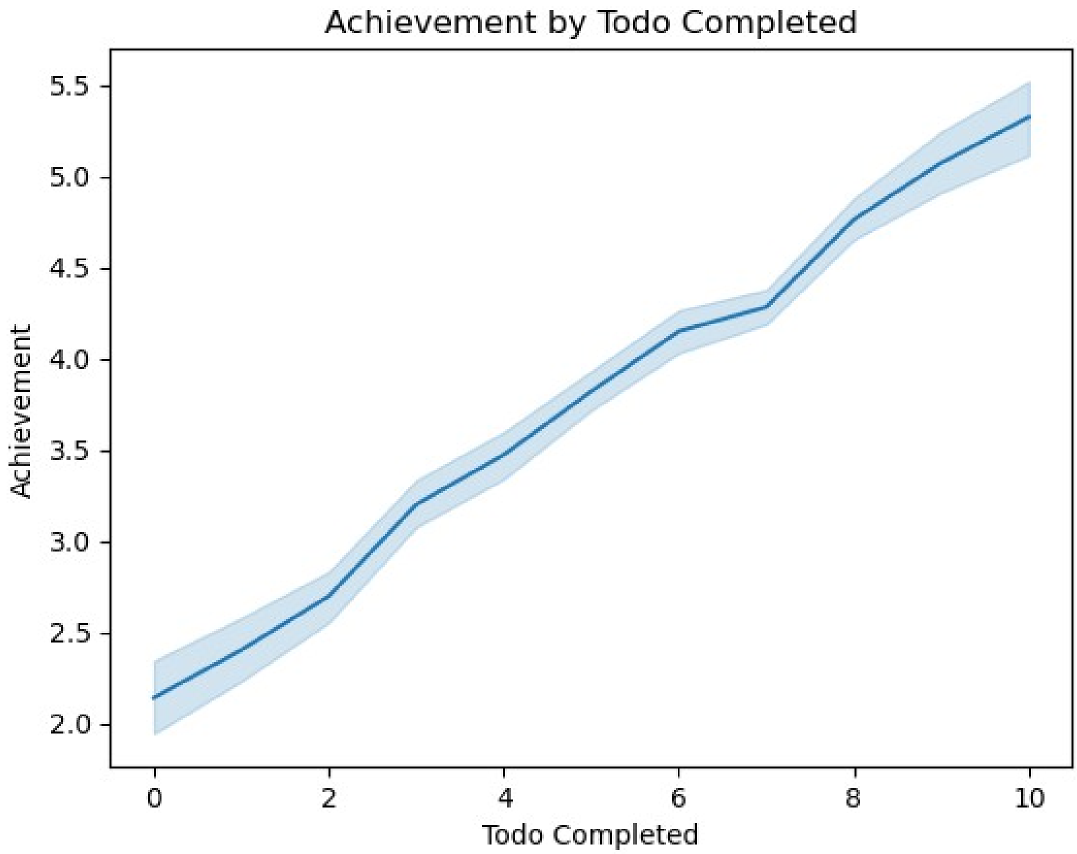

```
C:\Users\siddh\AppData\Local\Temp\ipykernel_5704\3776454138.py:7:
FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same
effect.

  sns.lineplot(x='AGE', y='ACHIEVEMENT', hue='GENDER', data=df,
ci=None)  # ci=None removes confidence intervals
c:\Users\siddh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\siddh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



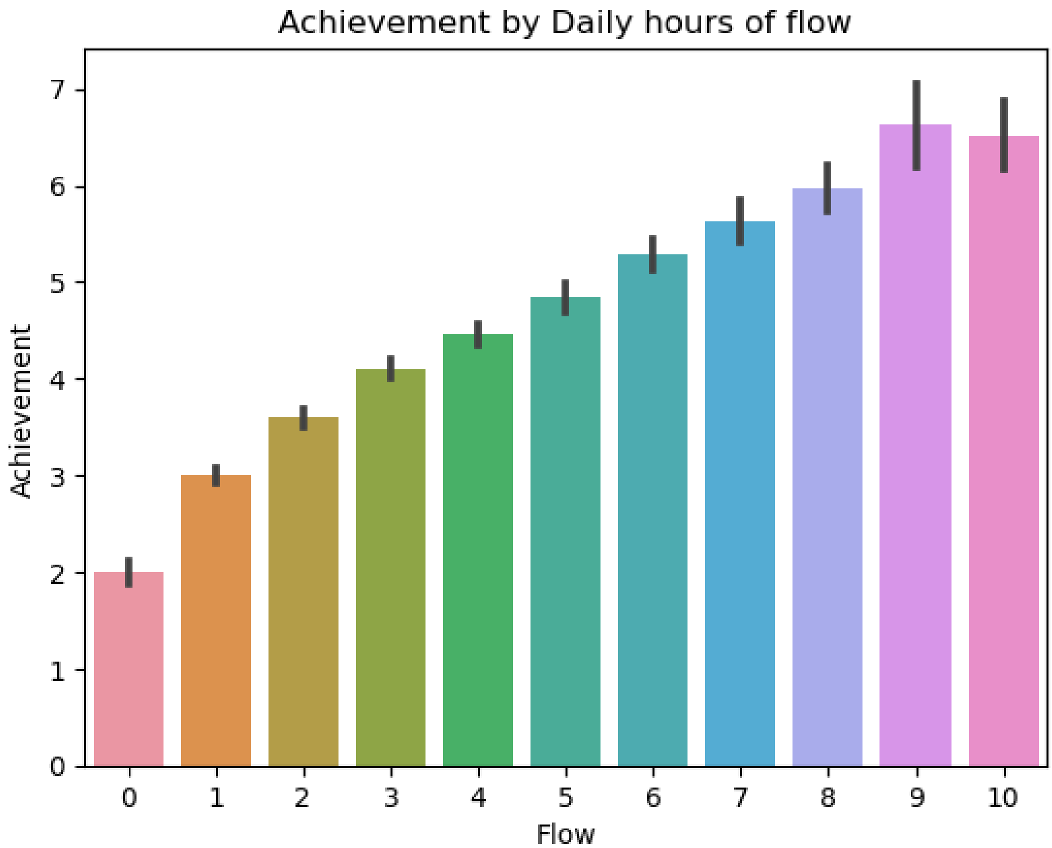

```
sns.barplot(x='TODO_COMPLETED', y='ACHIEVEMENT', data=df)
plt.title('Achievement by Todo Completed')
plt.xlabel('Todo Completed')
plt.ylabel('Achievement')
plt.show()

sns.lineplot(x='TODO_COMPLETED', y='ACHIEVEMENT', data=df)
plt.title('Achievement by Todo Completed')
plt.xlabel('Todo Completed')
plt.ylabel('Achievement')
plt.show()
```



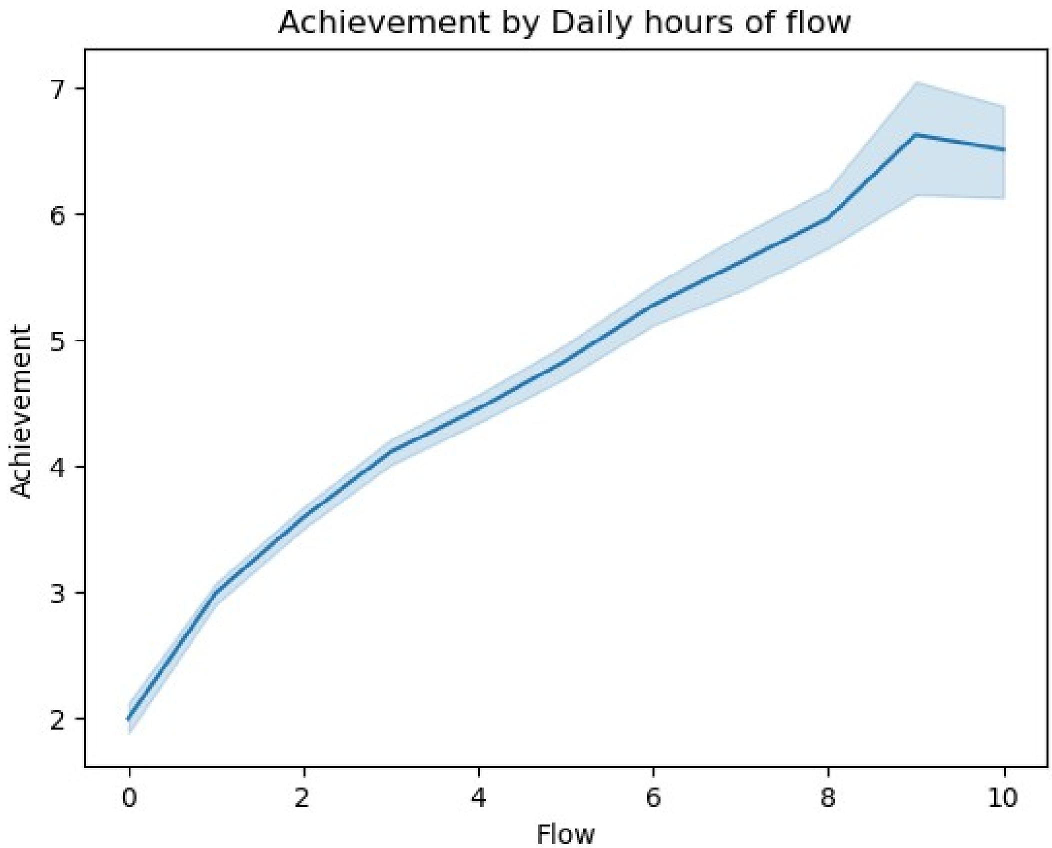

```
c:\Users\siddh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\siddh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



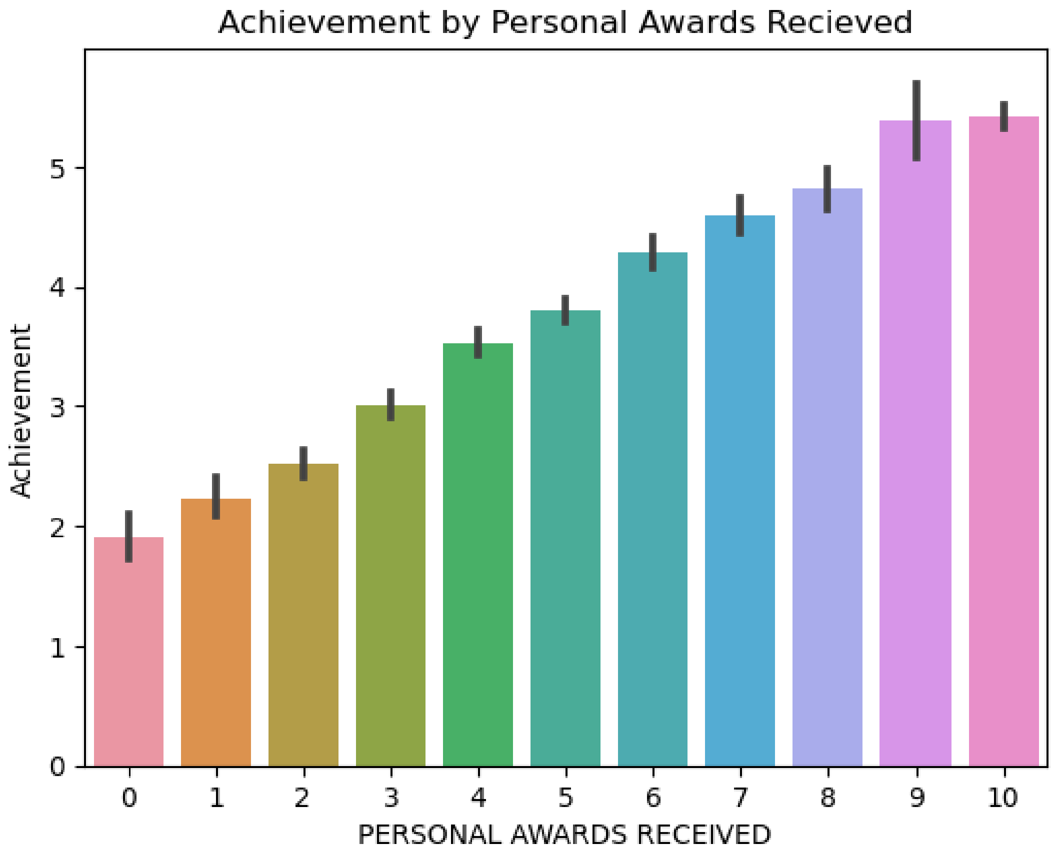

```
sns.barplot(x='FLOW', y='ACHIEVEMENT', data=df)
plt.title('Achievement by Daily hours of flow')
plt.xlabel('Flow')
plt.ylabel('Achievement')
plt.show()

sns.lineplot(x='FLOW', y='ACHIEVEMENT', data=df)
plt.title('Achievement by Daily hours of flow')
plt.xlabel('Flow')
plt.ylabel('Achievement')
plt.show()
```



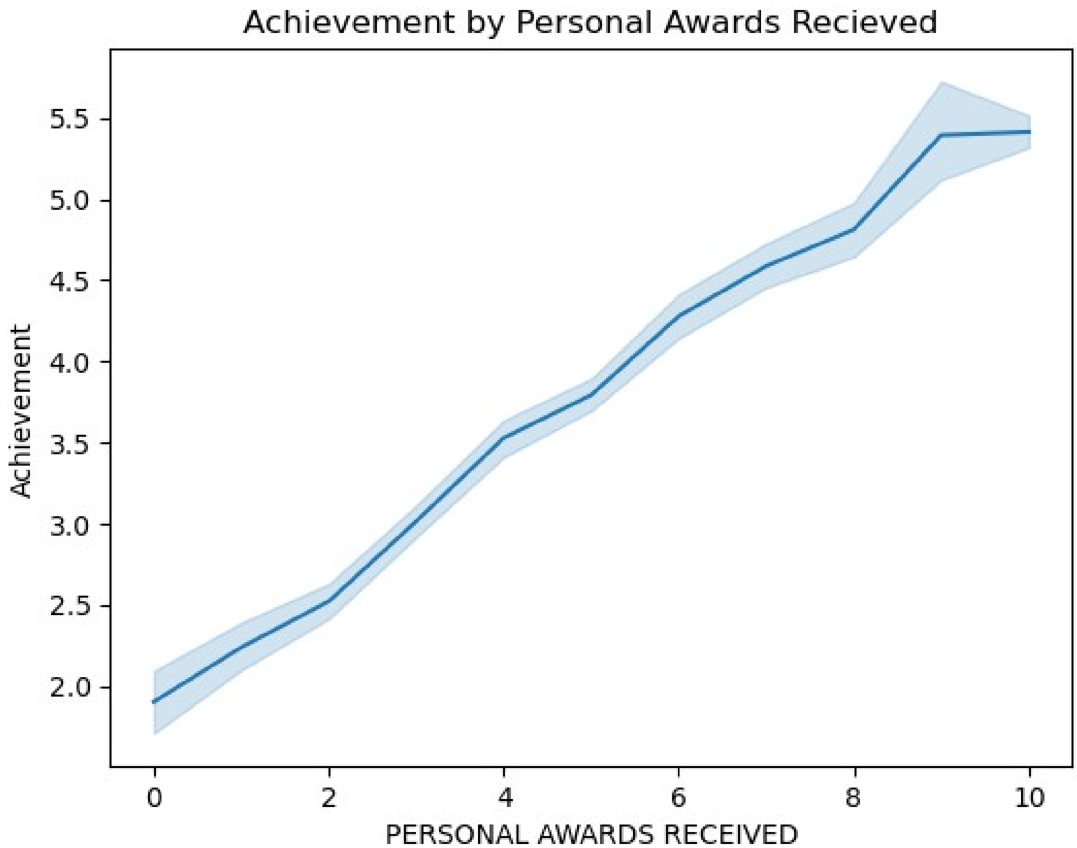

```
c:\Users\siddh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\siddh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
sns.barplot(x='PERSONAL_AWARDS', y='ACHIEVEMENT', data=df)
plt.title('Achievement by Personal Awards Recieved')
plt.xlabel('PERSONAL AWARDS RECEIVED')
plt.ylabel('Achievement')
plt.show()

sns.lineplot(x='PERSONAL_AWARDS', y='ACHIEVEMENT', data=df)
plt.title('Achievement by Personal Awards Recieved')
plt.xlabel('PERSONAL AWARDS RECEIVED')
plt.ylabel('Achievement')
plt.show()
```

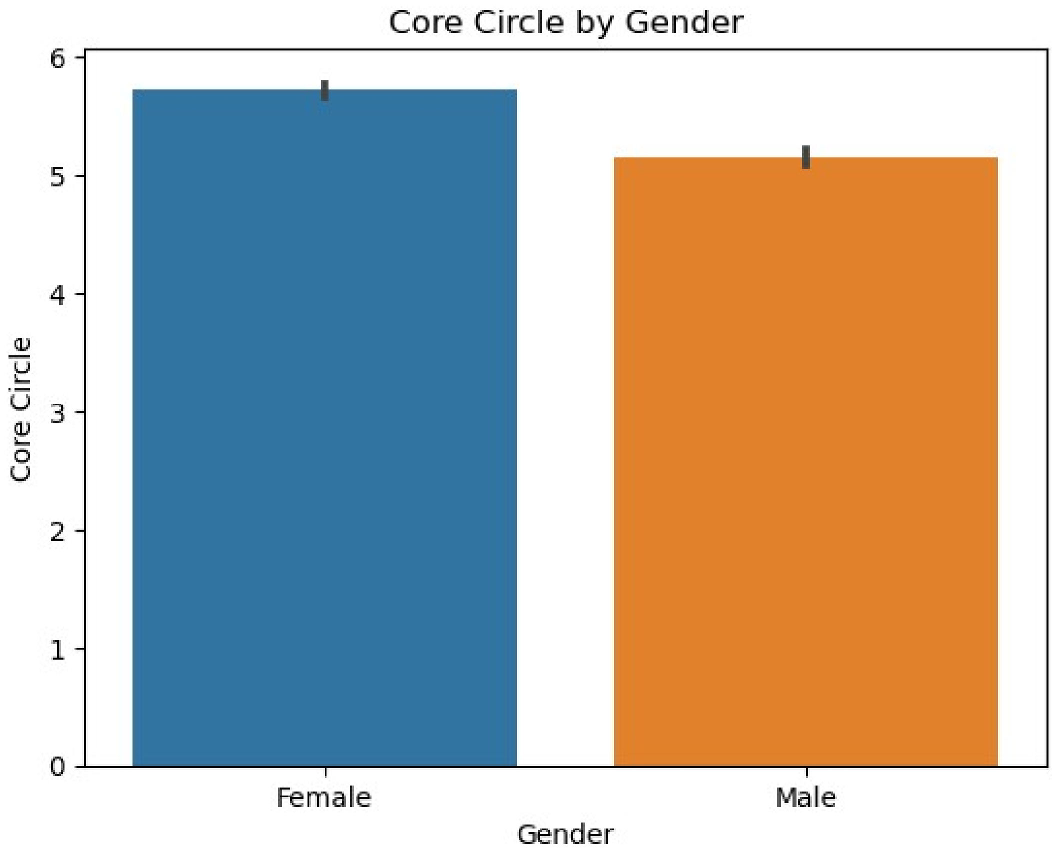



```
c:\Users\siddh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\siddh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

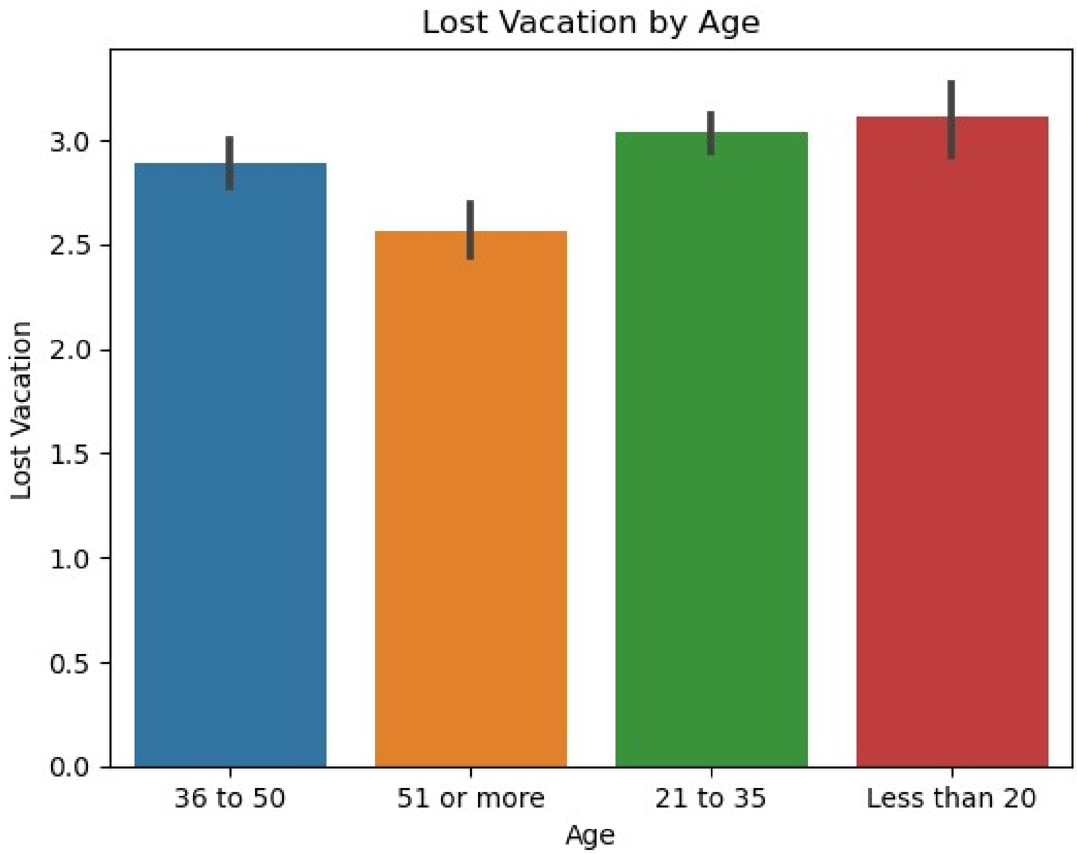


Conclusions: Woman reports slightly more personal achievements in their early age while men report more after age 36. Our daily productivity, the ability to flow hroughtout the day and personal awards such as diploma and other certificates all contribute to higher levels of personal achievements.

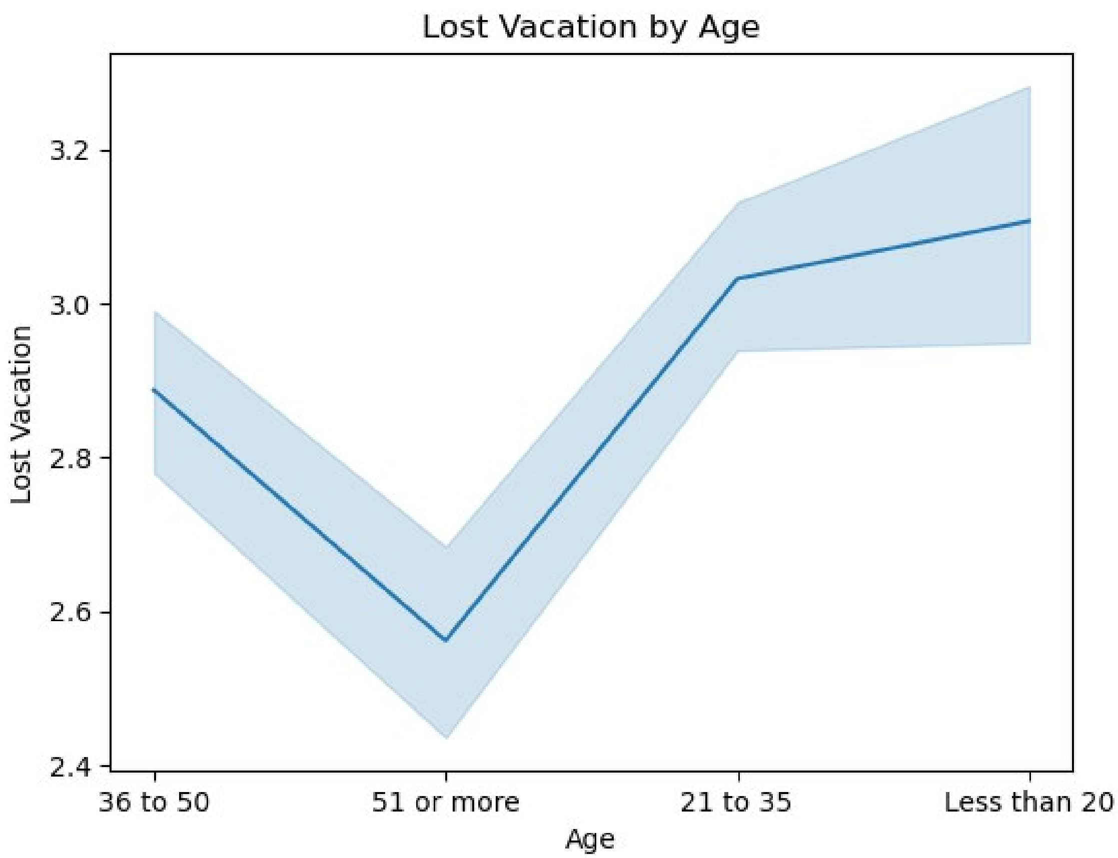
```
Analysis of factors correlating to Connection

sns.barplot(x='GENDER', y='CORE_CIRCLE', data=df)
plt.title('Core Circle by Gender')
plt.xlabel('Gender')
plt.ylabel('Core Circle')
plt.show()
```




```
sns.barplot(x='AGE', y='LOST_VACATION', data=df)
plt.title('Lost Vacation by Age')
plt.xlabel('Age')
plt.ylabel('Lost Vacation')
plt.show()

sns.lineplot(x='AGE', y='LOST_VACATION', data=df)
plt.title('Lost Vacation by Age')
plt.xlabel('Age')
plt.ylabel('Lost Vacation')
plt.show()
```



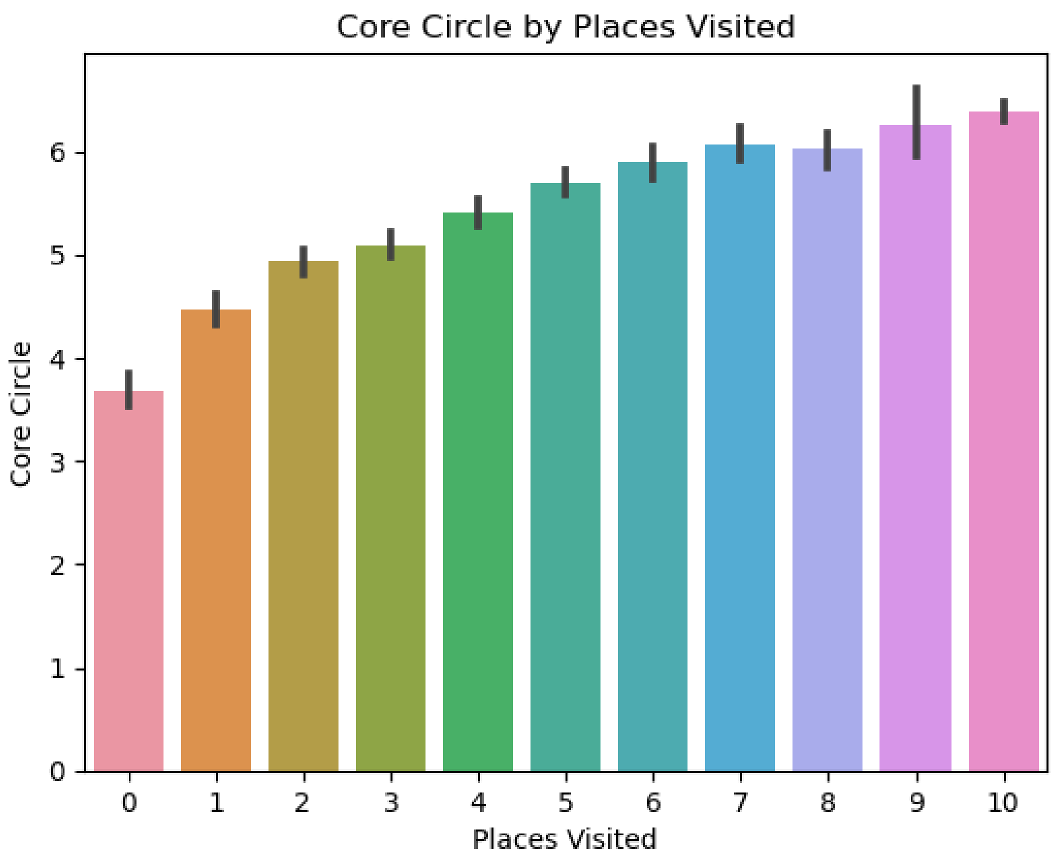

```
c:\Users\siddh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\siddh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



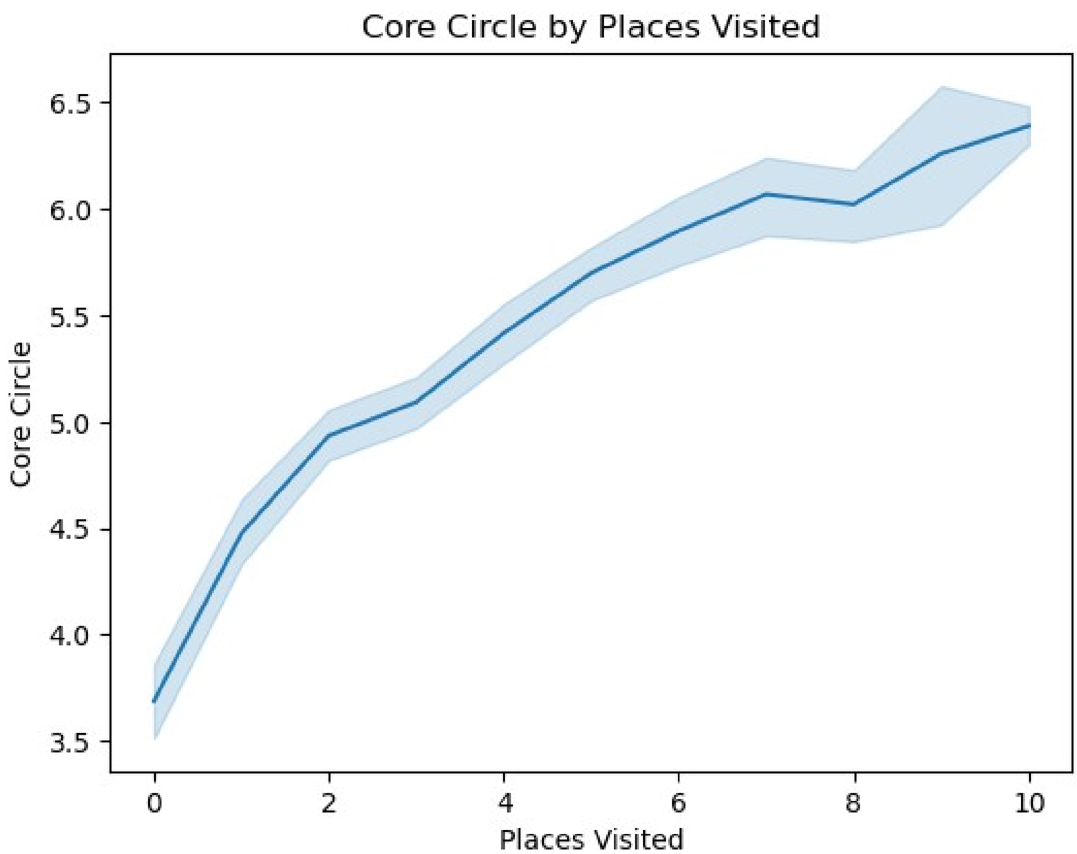

```
sns.barplot(x='LOST_VACATION', y='DAILY_STRESS', data=df)
plt.title('Daily Stress by Lost Vacation')
plt.xlabel('Lost Vacation')
plt.ylabel('Daily Stress')
plt.show()

sns.lineplot(x='LOST_VACATION', y='DAILY_STRESS', data=df)
plt.title('Daily Stress by Lost Vacation')
plt.xlabel('Lost Vacation')
plt.ylabel('Daily Stress')
plt.show()
```


```
c:\Users\siddh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\siddh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

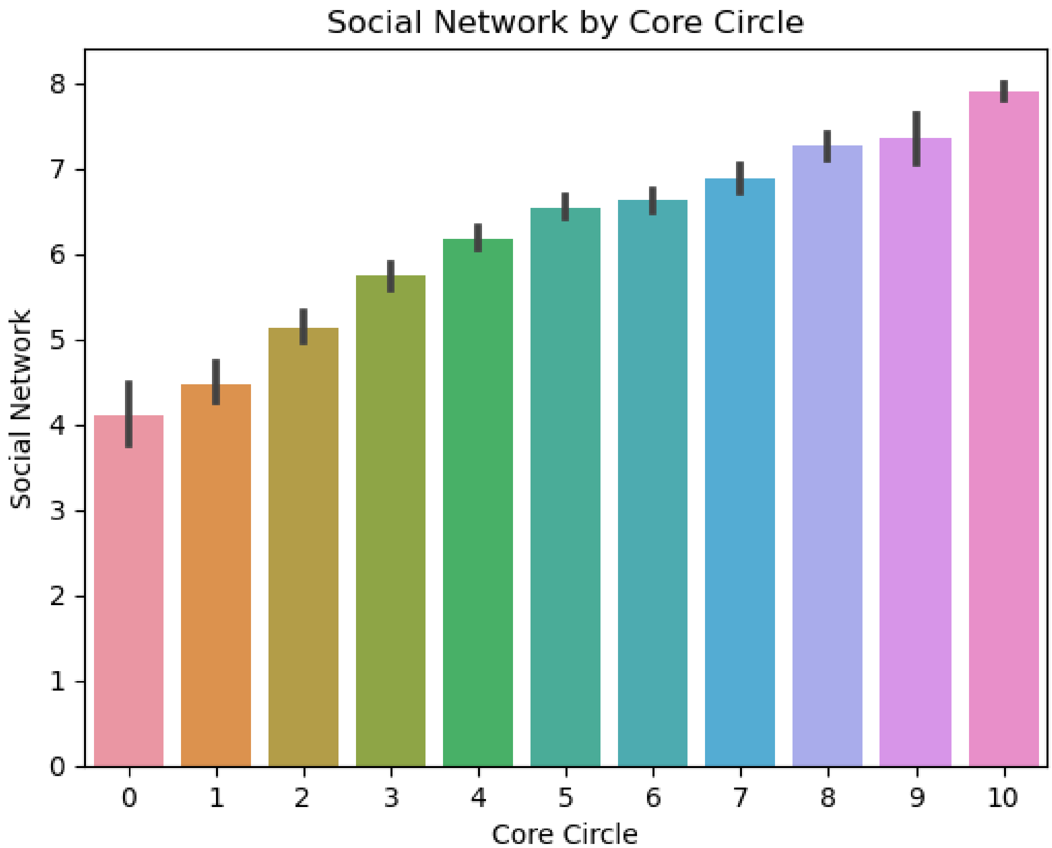

```
sns.barplot(x='PLACES_VISITED', y='CORE_CIRCLE', data=df)
plt.title('Core Circle by Places Visited')
plt.xlabel('Places Visited')
plt.ylabel('Core Circle')
plt.show()

sns.lineplot(x='PLACES_VISITED', y='CORE_CIRCLE', data=df)
plt.title('Core Circle by Places Visited')
plt.xlabel('Places Visited')
plt.ylabel('Core Circle')
plt.show()
```



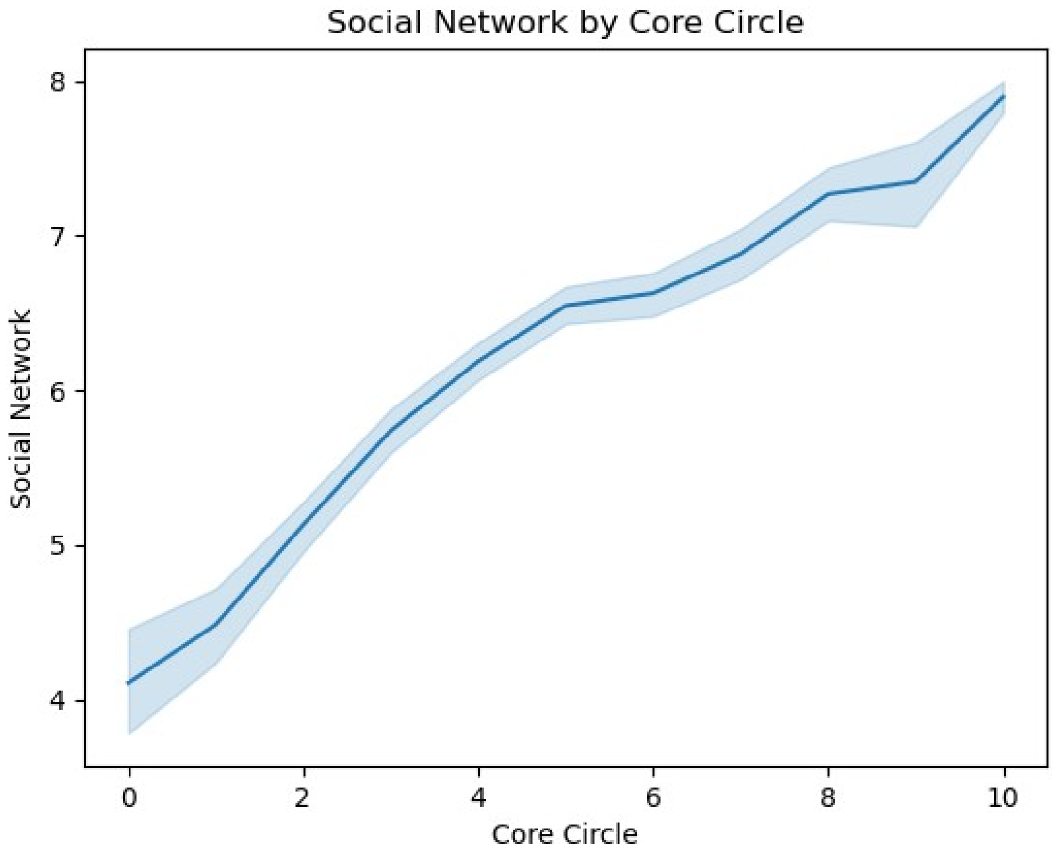

```
c:\Users\siddh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\siddh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
sns.barplot(x='CORE_CIRCLE', y='SOCIAL_NETWORK', data=df)
plt.title('Social Network by Core Circle')
plt.xlabel('Core Circle')
plt.ylabel('Social Network')
plt.show()

sns.lineplot(x='CORE_CIRCLE', y='SOCIAL_NETWORK', data=df)
plt.title('Social Network by Core Circle')
plt.xlabel('Core Circle')
plt.ylabel('Social Network')
plt.show()
```

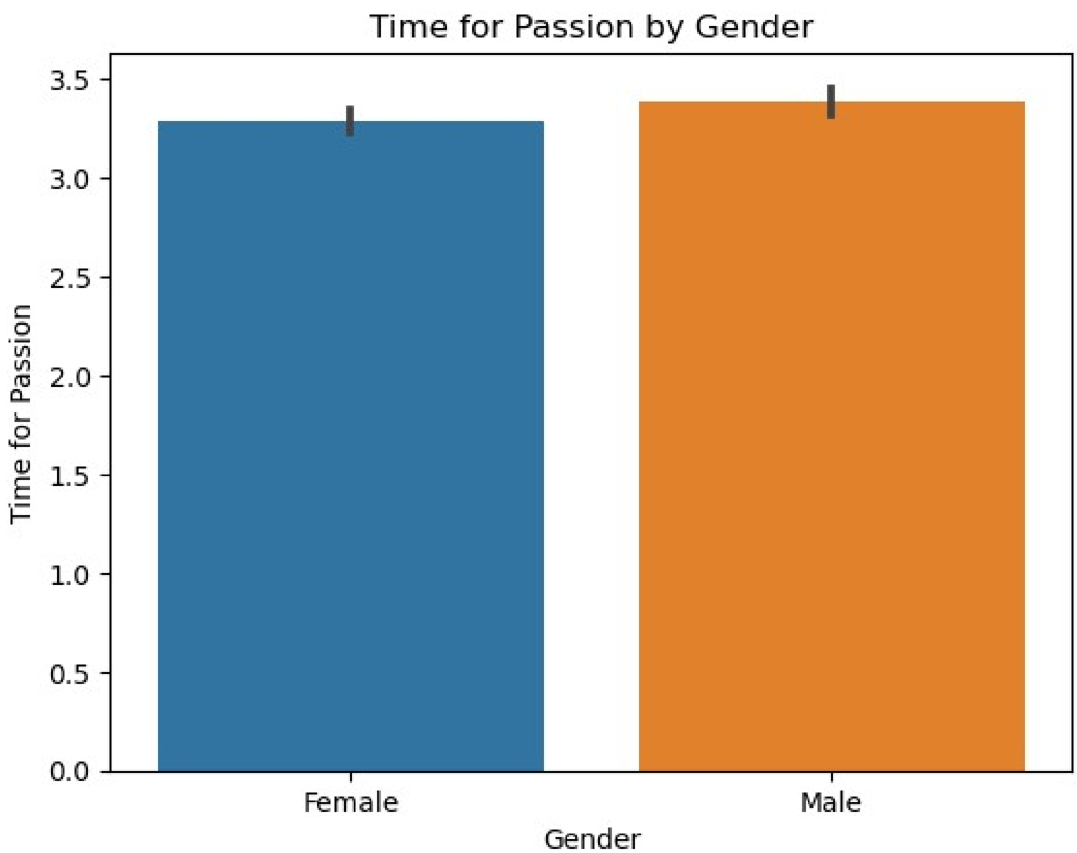



```
c:\Users\siddh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\siddh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

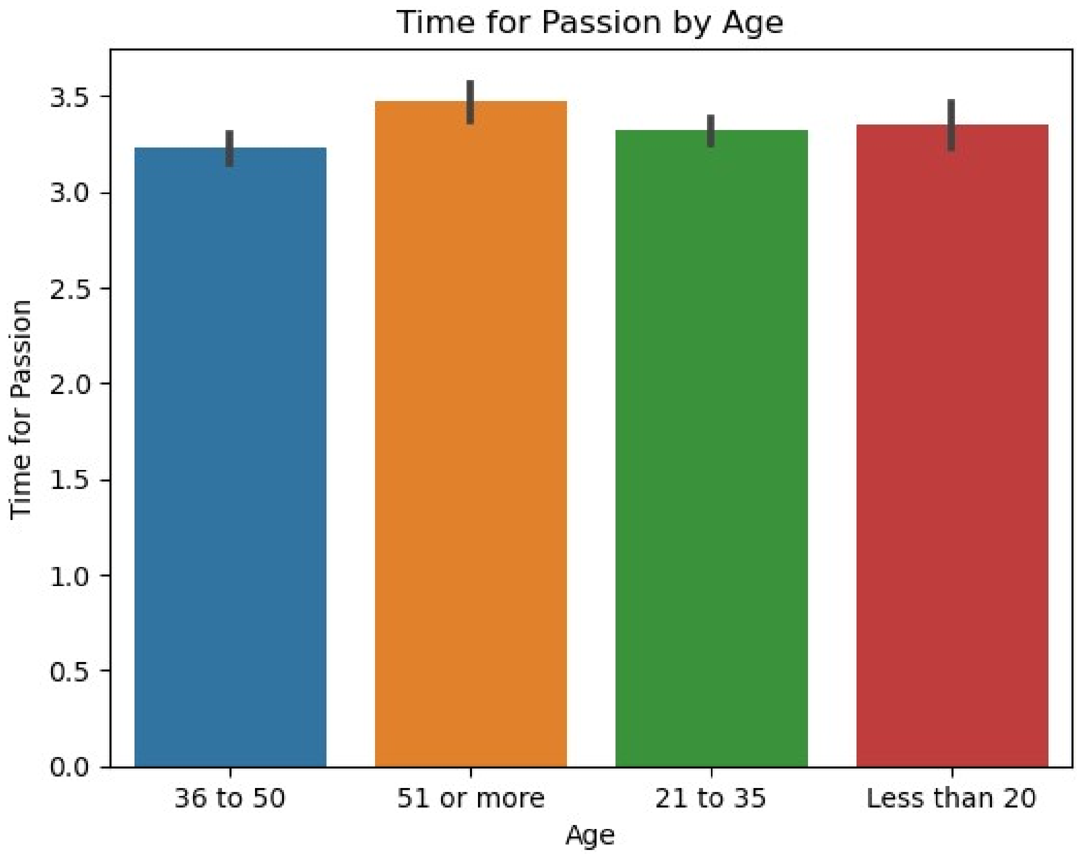


Conclusions: Womem appear to have a stronger circle of friends and family than men. People in the age group 21 to 35 forfeit a maximum of vacation days, when compared to other age groups. Overall, the level of their sress increase as we lose more vacation days. But there is a slight dip between 7 and 9 days for lost vacations, as if losing six or many more vacation days does not have any impact anymore on the stress level.

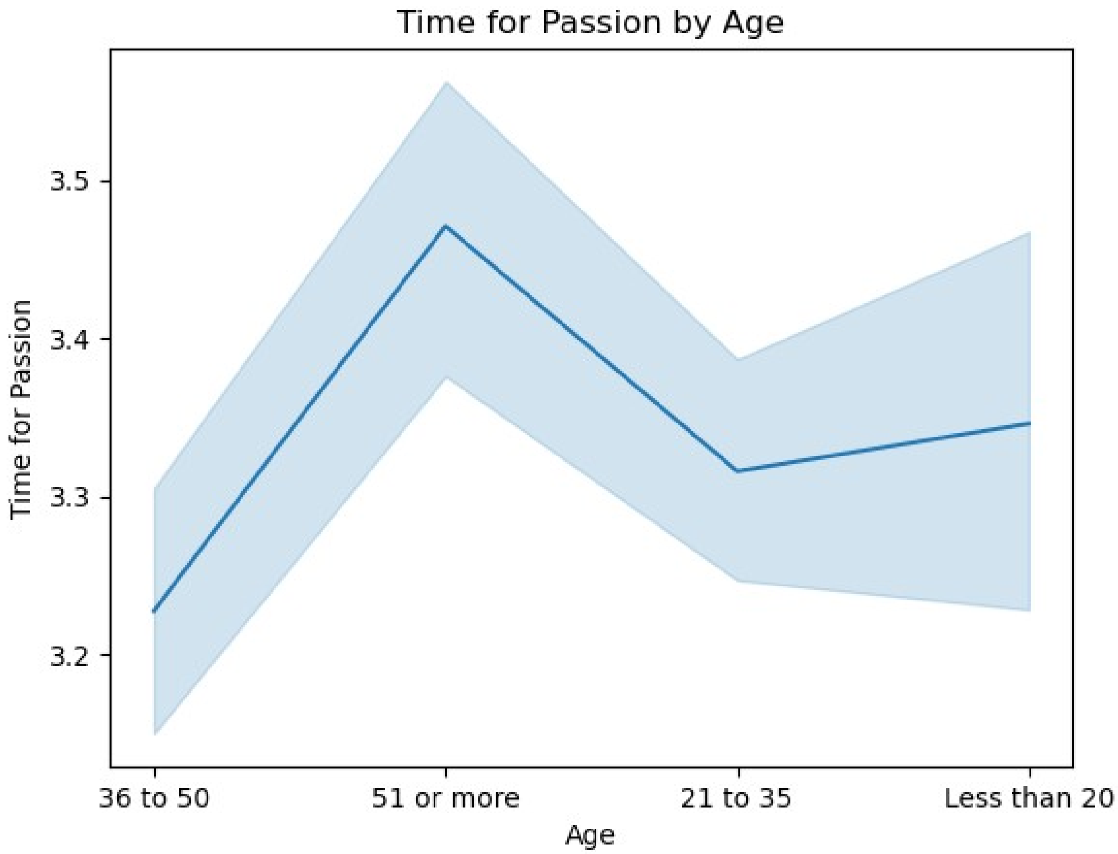
```
Analysis of factors correlating to Passion

sns.barplot(x='GENDER', y='TIME_FOR_PASSION', data=df)
plt.title('Time for Passion by Gender')
plt.xlabel('Gender')
plt.ylabel('Time for Passion')
plt.show()
```



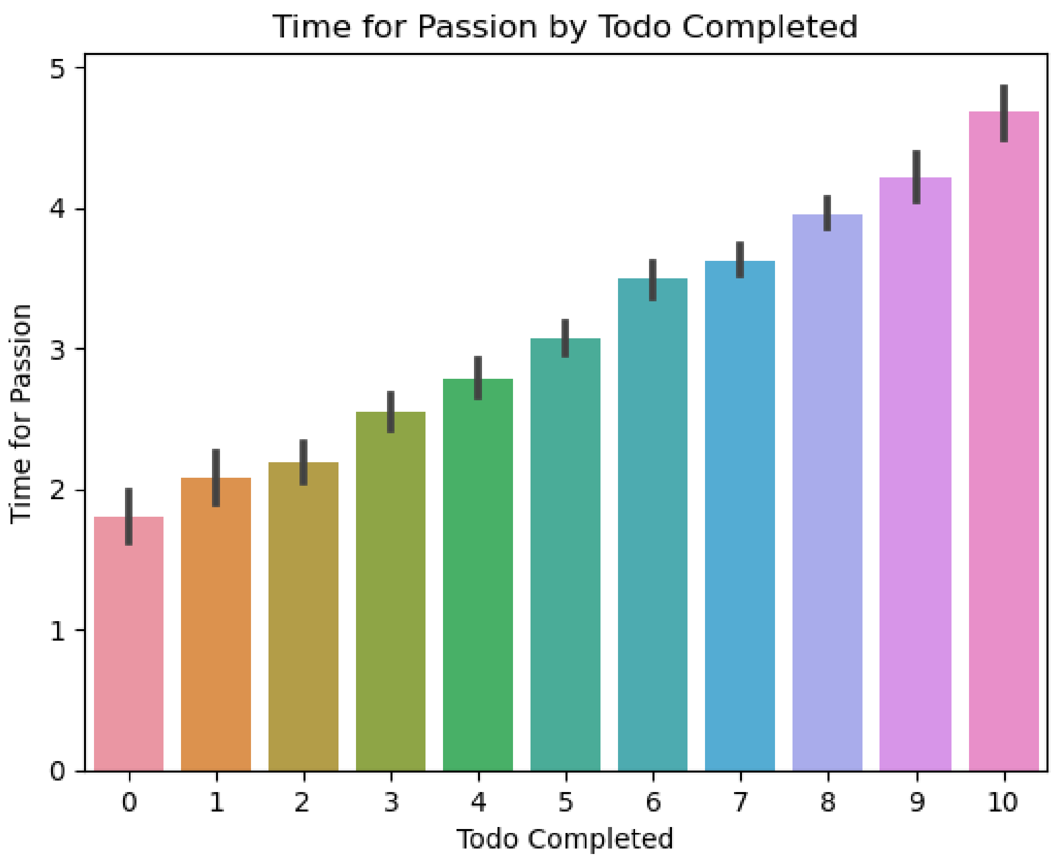

```
sns.barplot(x='AGE', y='TIME_FOR_PASSION', data=df)
plt.title('Time for Passion by Age')
plt.xlabel('Age')
plt.ylabel('Time for Passion')
plt.show()

sns.lineplot(x='AGE', y='TIME_FOR_PASSION', data=df)
plt.title('Time for Passion by Age')
plt.xlabel('Age')
plt.ylabel('Time for Passion')
plt.show()
```



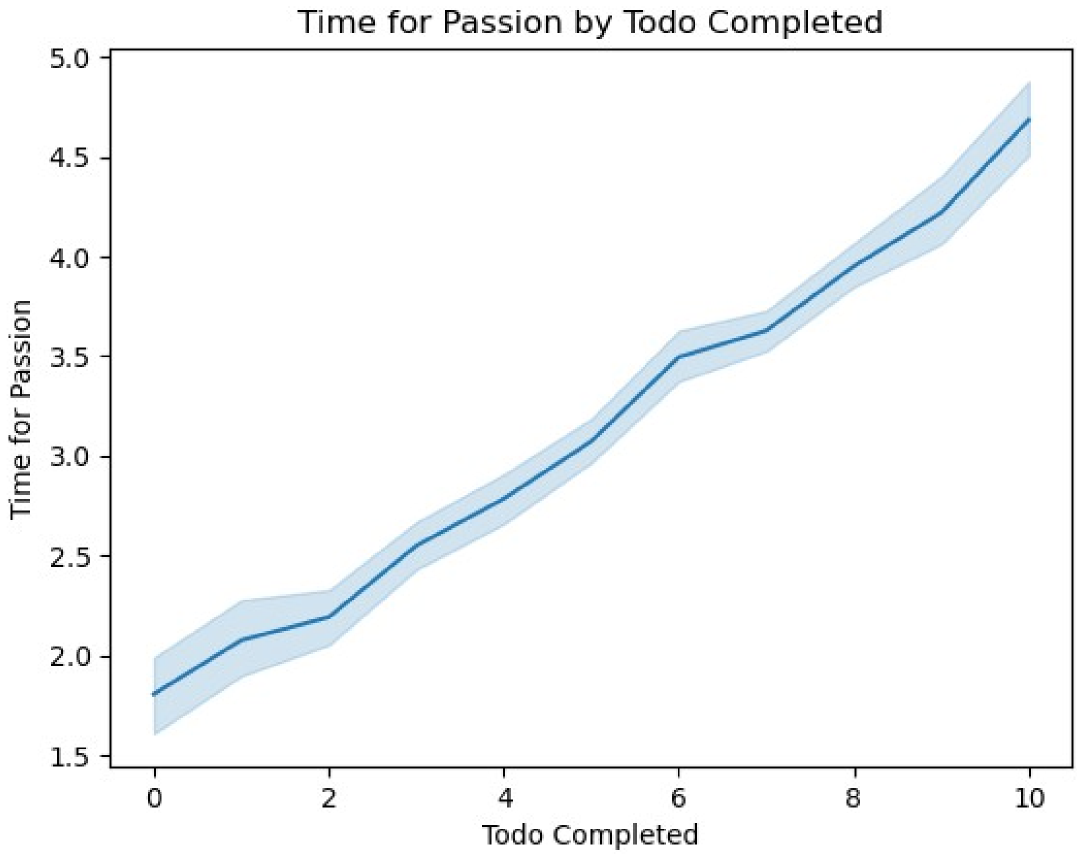

```
c:\Users\siddh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\siddh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



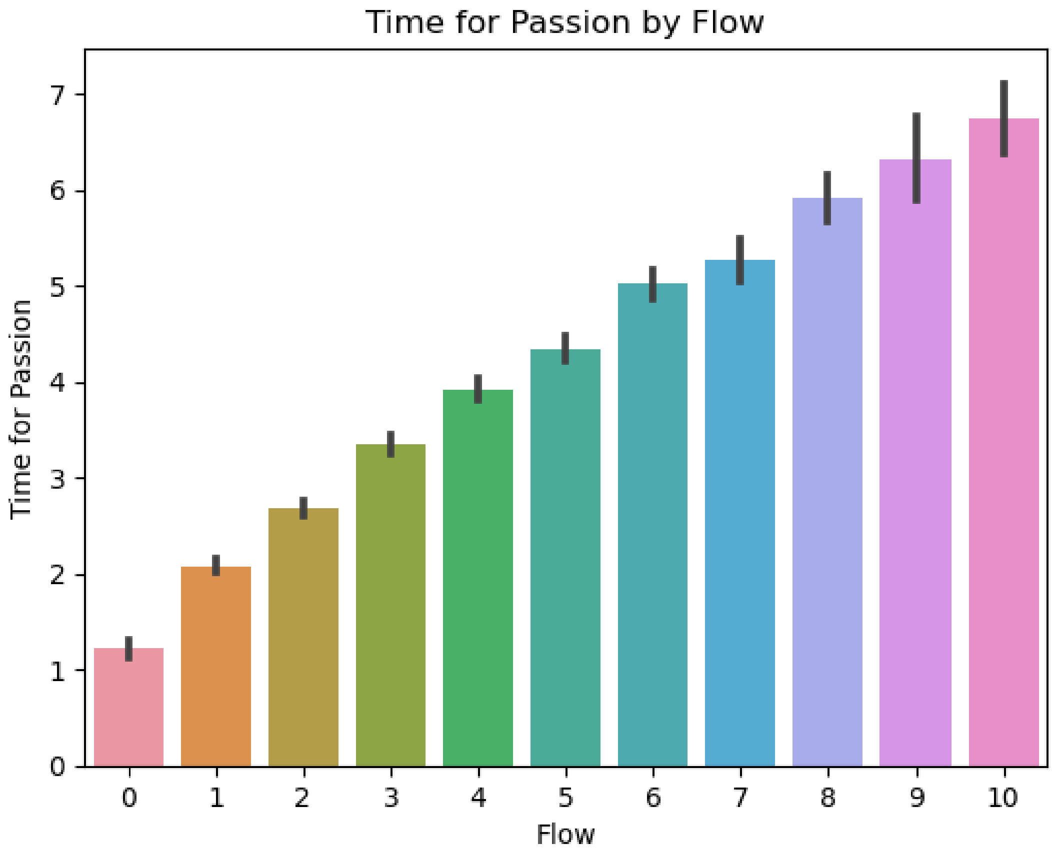

```
sns.barplot(x='TODO_COMPLETED', y='TIME_FOR_PASSION', data=df)
plt.title('Time for Passion by Todo Completed')
plt.xlabel('Todo Completed')
plt.ylabel('Time for Passion')
plt.show()

sns.lineplot(x='TODO_COMPLETED', y='TIME_FOR_PASSION', data=df)
plt.title('Time for Passion by Todo Completed')
plt.xlabel('Todo Completed')
plt.ylabel('Time for Passion')
plt.show()
```



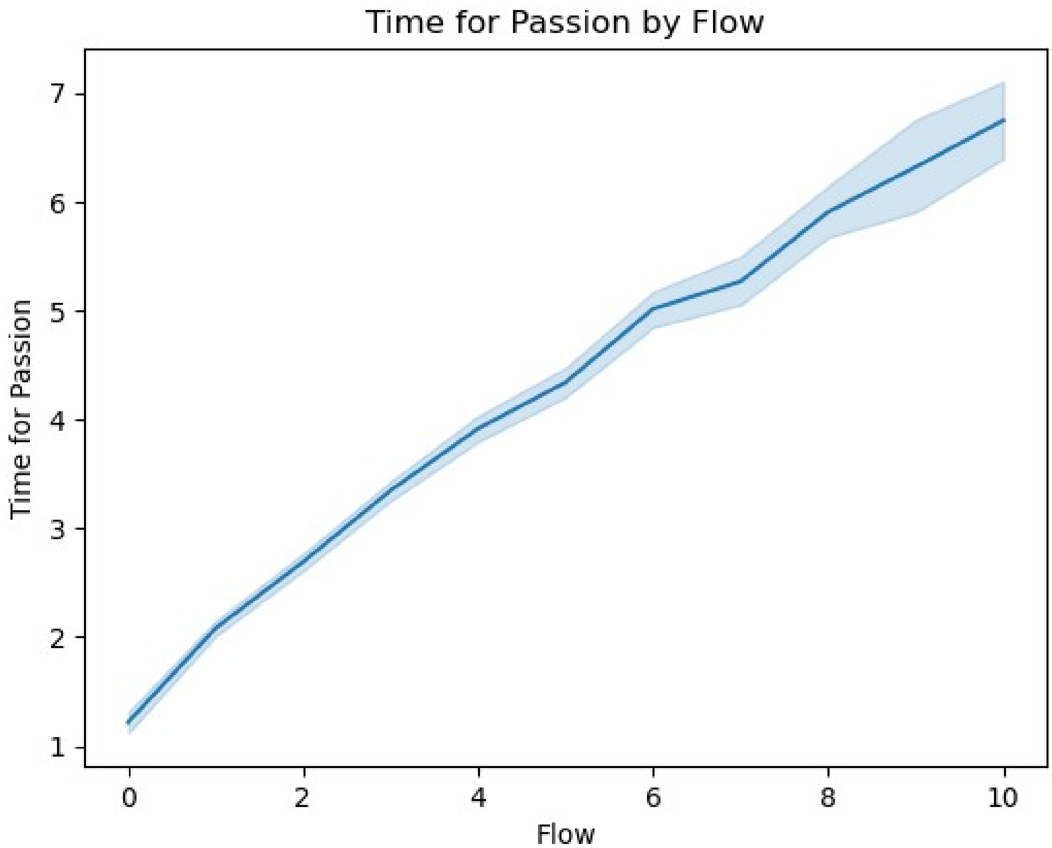

```
c:\Users\siddh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\siddh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



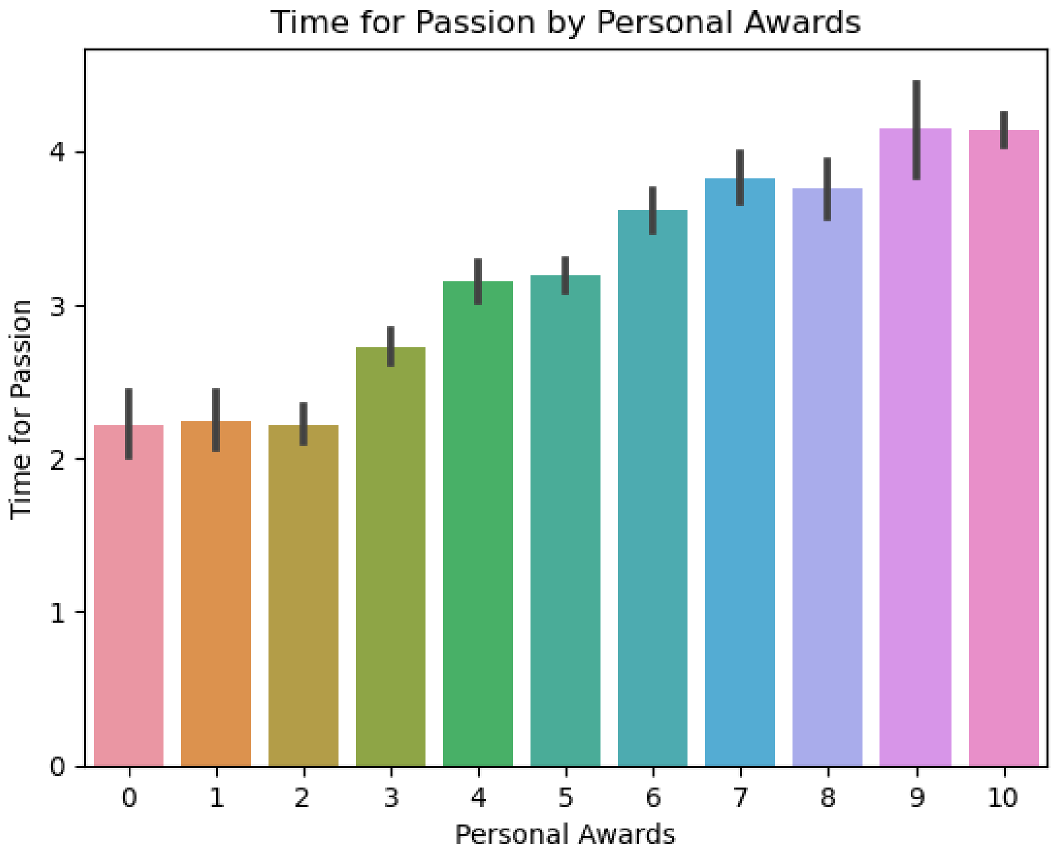

```
sns.barplot(x='FLOW', y='TIME_FOR_PASSION', data=df)
plt.title('Time for Passion by Flow')
plt.xlabel('Flow')
plt.ylabel('Time for Passion')
plt.show()

sns.lineplot(x='FLOW', y='TIME_FOR_PASSION', data=df)
plt.title('Time for Passion by Flow')
plt.xlabel('Flow')
plt.ylabel('Time for Passion')
plt.show()
```



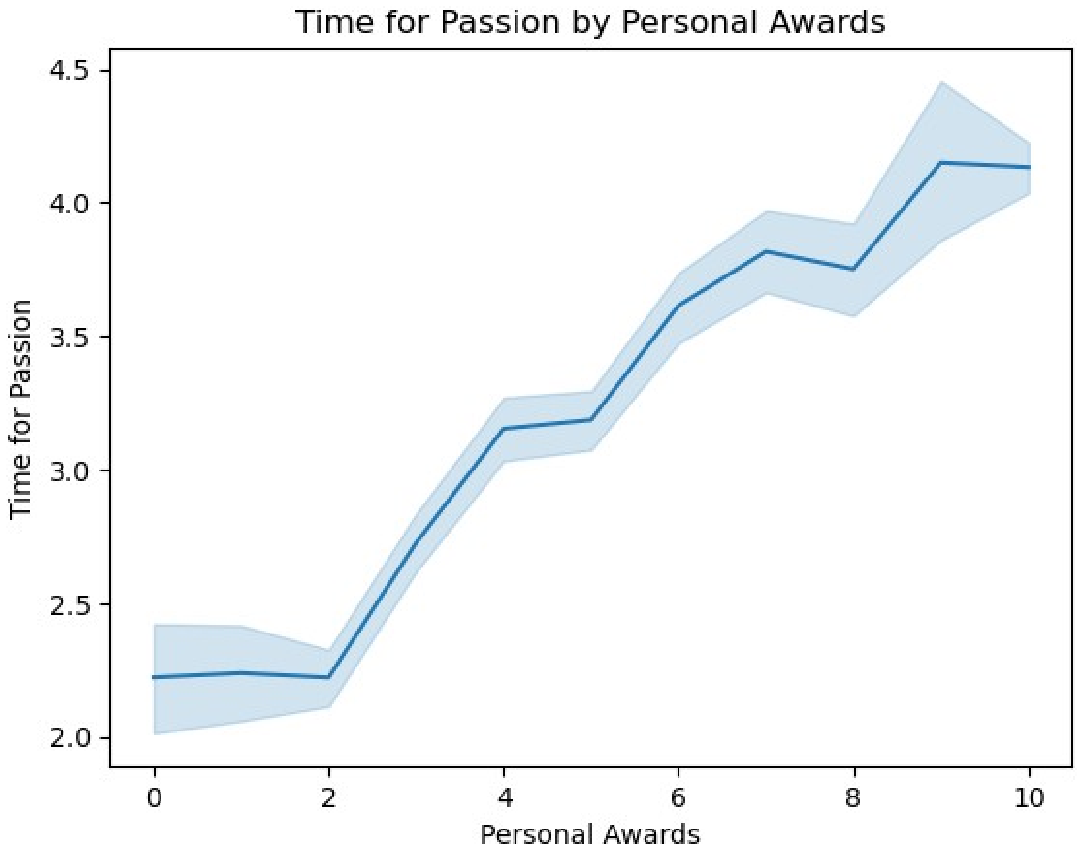

```
c:\Users\siddh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\siddh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```




```
sns.barplot(x='PERSONAL_AWARDS', y='TIME_FOR_PASSION', data=df)
plt.title('Time for Passion by Personal Awards')
plt.xlabel('Personal Awards')
plt.ylabel('Time for Passion')
plt.show()

sns.lineplot(x='PERSONAL_AWARDS', y='TIME_FOR_PASSION', data=df)
plt.title('Time for Passion by Personal Awards')
plt.xlabel('Personal Awards')
plt.ylabel('Time for Passion')
plt.show()
```




```
c:\Users\siddh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\siddh\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed
in a future version. Convert inf values to NaN before operating
instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



Conclusions: Men appear to find more time for their passion, especially in their younger and older ages. The three factors correlating the most with our ability to find time for our passions are: Our daily personal productivity Daily flow The personal awards we received

DONE BY

S SIDDHARTH 21BCE7284
N KEERTHIKA 22BCE9691
S RITHISH 21BCE8829
T NAGA CHARAN 21BCE7829
K RAM CHARAN 21BCE9236
H BALASUBRAMANI 21BCE8218
D V SRIDHARA REDDY 21BCE9241
T CHINMAYA GAYATHRI 22BCE7031
K ANSHU 22BCE9508