Aerofit Casestudy

Problem Statement

The AeroFit market research team aims to discern the distinct attributes of the target audience for each treadmill variant in the company's product. In order to enhance their ability to recommend the most suitable treadmills to prospective customers. The team has chosen to explore potential disparities in customer characteristics like Age,Gender,Income,Fitness etc across the various products

In [1]:
```
# Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Data Inspection & Basic Metrices

In [2]:
```
# Load Data
url='https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/125/orig
df = pd.read_csv(url)
```

In [3]:
```
df.head()
```

Out[3]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	KP281	18	Male	14	Single	3	4	29562	112
1	KP281	19	Male	15	Single	2	3	31836	75
2	KP281	19	Female	14	Partnered	4	3	30699	66
3	KP281	19	Male	12	Single	3	3	32973	85
4	KP281	20	Male	13	Partnered	4	2	35247	47

In [4]:
```
df.shape
```

Out[4]:
```
(180, 9)
```

In [5]:
```
df.dtypes
```

Out[5]:
```
Product          object
Age               int64
Gender           object
Education         int64
MaritalStatus    object
Usage             int64
Fitness           int64
Income            int64
Miles             int64
dtype: object
```

Observations 💡

```
1. There is a Chance to Convert 'object' to 'Category' for the
following columns
    a. Gender
    b. MaritalStatus
But , We have Small Memory Usage so it is not affect any Memory
Usage in large scale
```

In [6]:
```
df.nunique()
```

Out[6]:
```
Product           3
Age              32
Gender            2
Education         8
MaritalStatus     2
Usage             6
Fitness           5
Income           62
Miles            37
dtype: int64
```

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180 entries, 0 to 179
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   Product        180 non-null    object
 1   Age            180 non-null    int64 
 2   Gender         180 non-null    object
 3   Education      180 non-null    int64 
 4   MaritalStatus  180 non-null    object
 5   Usage          180 non-null    int64 
 6   Fitness        180 non-null    int64 
 7   Income         180 non-null    int64 
 8   Miles          180 non-null    int64 
dtypes: int64(6), object(3)
memory usage: 12.8+ KB
```

In [8]:
```
# Null Check
df.isnull().sum()
```

Out[8]:
```
Product          0
Age              0
Gender           0
Education        0
MaritalStatus    0
Usage            0
Fitness          0
Income           0
Miles            0
dtype: int64
```

Observations 💡

```
No NULL Values
```

Statistical Summary

In [9]:
```
np.round(df.describe())
```

Out[9]:

	Age	Education	Usage	Fitness	Income	Miles
count	180.0	180.0	180.0	180.0	180.0	180.0
mean	29.0	16.0	3.0	3.0	53720.0	103.0
std	7.0	2.0	1.0	1.0	16507.0	52.0
min	18.0	12.0	2.0	1.0	29562.0	21.0
25%	24.0	14.0	3.0	3.0	44059.0	66.0
50%	26.0	16.0	3.0	3.0	50596.0	94.0
75%	33.0	16.0	4.0	4.0	58668.0	115.0
max	50.0	21.0	7.0	5.0	104581.0	360.0

Observations 💡

```
1. Age       : Minimum Age is 18 & 50% of people below than Mean
which is 2️⃣9️⃣
2. Education : Avg Education is High School Standard (16 Yrs) &
Most of the People are Undergraduate Level
3. Usage     : Avg People Use treadmill trice in Week
4. Fitness   : Most of the Users Having Above Average Fitness
Level (75%)
5. Income    : Maximum No of People Earning > 1L
6. Miles     : Veteran People (Age - 50) are Running more than 3
times to the Mean Value (103)
```

In [10]:
```
df.describe(include='object')
```

Out[10]:

	Product	Gender	MaritalStatus
count	180	180	180
unique	3	2	2
top	KP281	Male	Partnered
freq	80	104	107

Observations 💡

```
1. KP281 is Used by 44% People
2. There is Huge Majority of Male people with 57%
3. 60% Married People are Using Aerofit Services
```

Non-Graphical Analysis

In [11]:

```
for i in range(df.shape[1]):
    print(df.columns[i]+':')
    print(df[df.columns[i]].unique(),df[df.columns[i]].nunique())
    print("************")
```

```
Product:
['KP281' 'KP481' 'KP781'] 3
************
Age:
[18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
 43 44 46 47 50 45 48 42] 32
************
Gender:
['Male' 'Female'] 2
************
Education:
[14 15 12 13 16 18 20 21] 8
************
MaritalStatus:
['Single' 'Partnered'] 2
************
Usage:
[3 2 4 5 6 7] 6
************
Fitness:
[4 3 2 1 5] 5
************
Income:
[ 29562  31836  30699  32973  35247  37521  36384  38658  40932  34110
  39795  42069  44343  45480  46617  48891  53439  43206  52302  51165
  50028  54576  68220  55713  60261  67083  56850  59124  61398  57987
  64809  47754  65220  62535  48658  54781  48556  58516  53536  61006
  57271  52291  49801  62251  64741  70966  75946  74701  69721  83416
  88396  90886  92131  77191  52290  85906 103336  99601  89641  95866
 104581  95508] 62
************
Miles:
[112  75  66  85  47 141 103  94 113  38 188  56 132 169  64  53 106  95
 212  42 127  74 170  21 120 200 140 100  80 160 180 240 150 300 280 260
 360] 37
************
```

Missing Value & Outlier Detection

In [12]:
```python
# Null Check
df.isnull().sum()
```

Out[12]:
```
Product          0
Age              0
Gender           0
Education        0
MaritalStatus    0
Usage            0
Fitness          0
Income           0
Miles            0
dtype: int64
```

Observations 💡

    No Null Values & Missing Values

Outliers

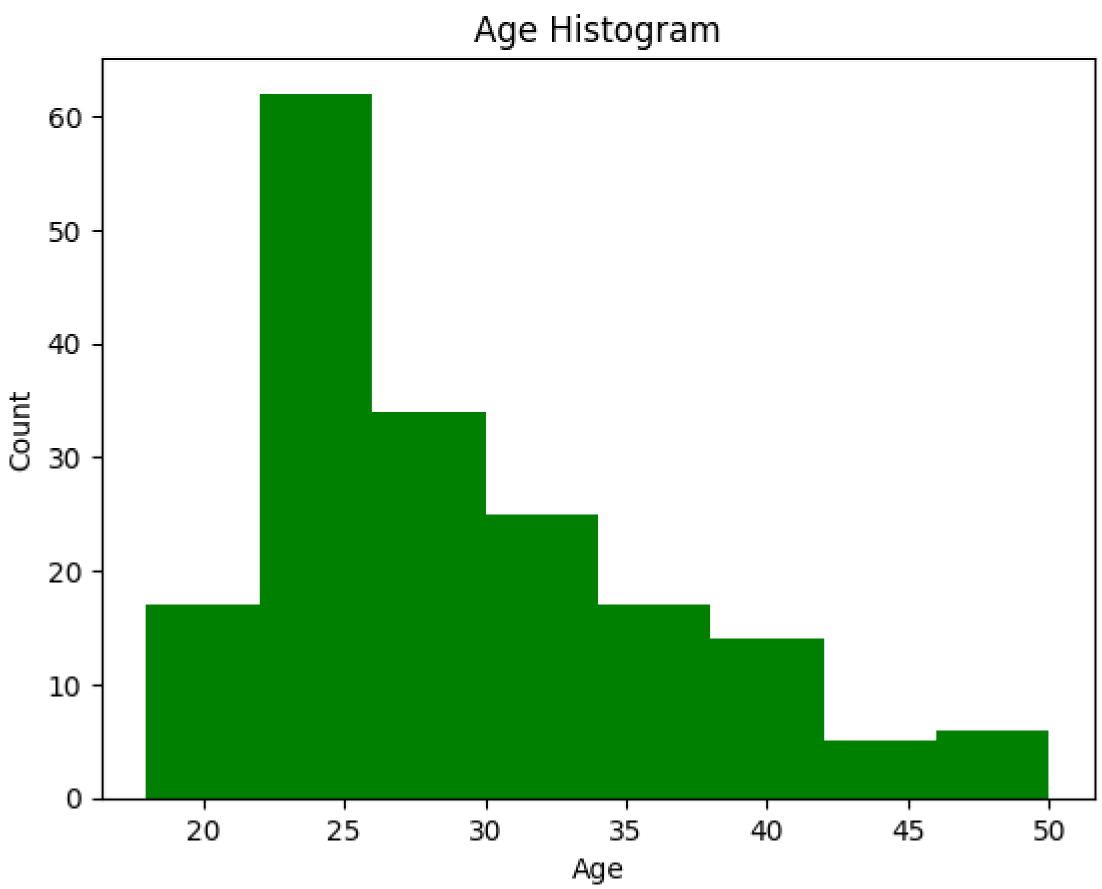
In [13]:
```
for i in range(df.shape[1]):
    if df[df.columns[i]].dtype=='int64':
        data = df[df.columns[i]]
        q1 = np.percentile(data, 25)
        q3 = np.percentile(data, 75)
        iqr = q3 - q1
        lower_bound = q1 - 1.5 * iqr
        upper_bound = q3 + 1.5 * iqr
        outliers = [x for x in data if x < lower_bound or x > upper_bound]
        print(f"{df.columns[i]} Outliers:", outliers)
        print()
```

```
Age Outliers: [47, 50, 48, 47, 48]

Education Outliers: [20, 21, 21, 21]

Usage Outliers: [6, 6, 6, 7, 6, 7, 6, 6, 6]

Fitness Outliers: [1, 1]

Income Outliers: [83416, 88396, 90886, 92131, 88396, 85906, 90886, 103336, 99601,
89641, 95866, 92131, 92131, 104581, 83416, 89641, 90886, 104581, 95508]

Miles Outliers: [188, 212, 200, 200, 200, 240, 300, 280, 260, 200, 360, 200, 200]
```

Visual Analysis

Univariate

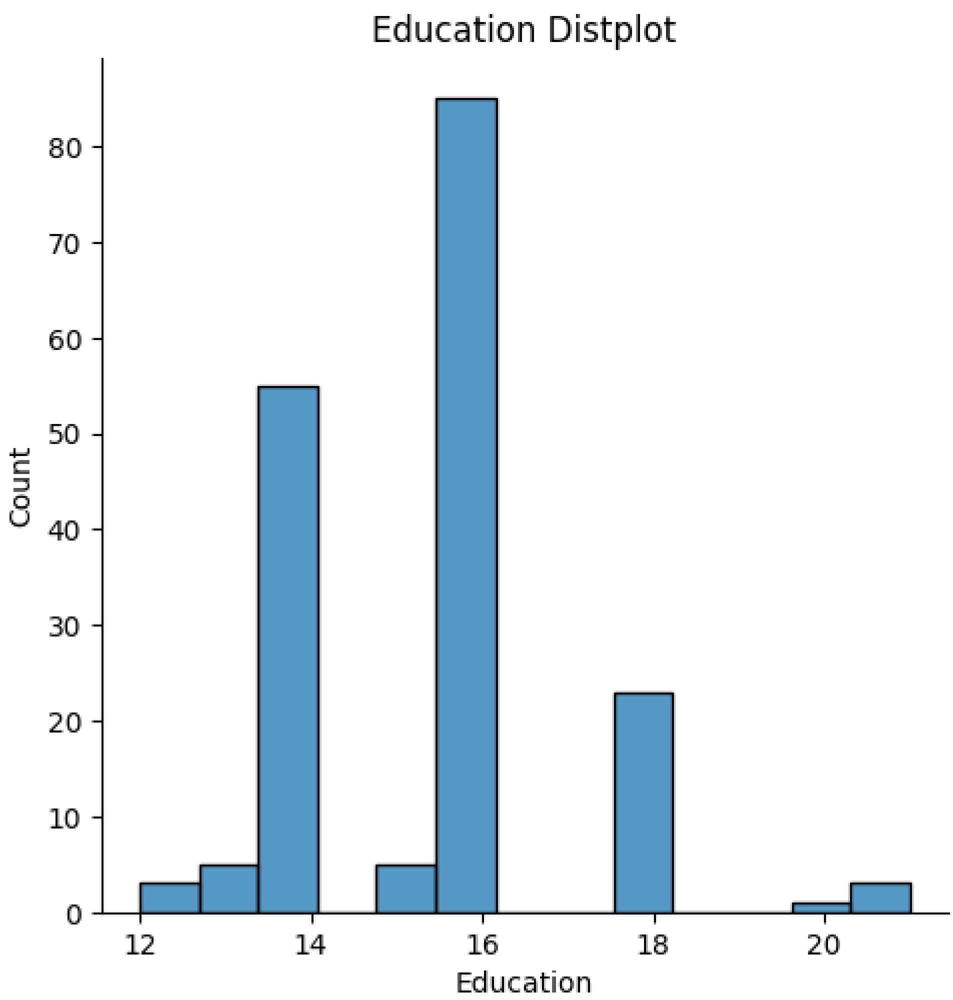
In [14]:
```
plt.hist(df['Age'],bins=8,color='Green')
plt.xlabel('Age')
plt.ylabel('Count')
plt.title('Age Histogram')
plt.show()
```



Observations 💡

```
1. There is a countinously downtrend with respect to age
2. Maximum People of Age group between 20-25 (Young People)
```

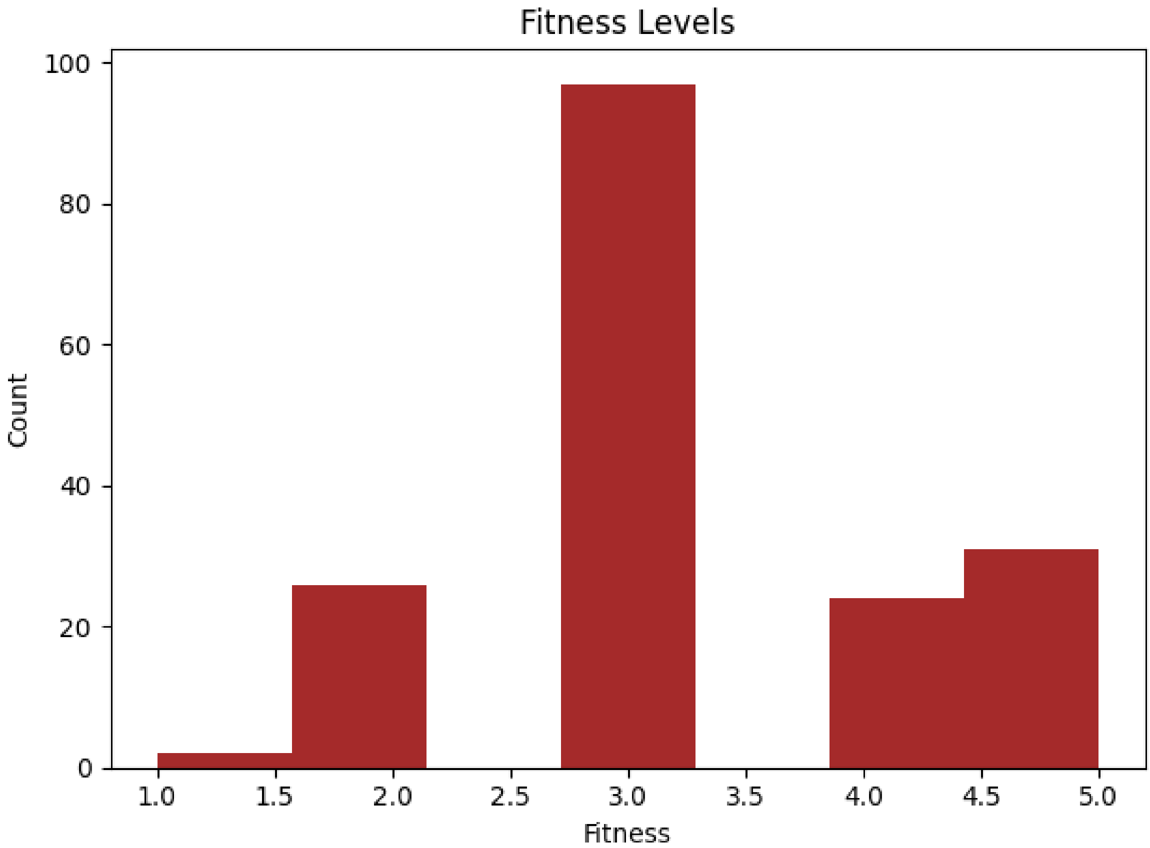
In [15]:
```
sns.displot(data = df['Education'])
plt.title('Education Distplot')
plt.show()
```



Observations 💡

```
77% Of People are Completed upto Pre University Education

    a.  30% of People Completed their High School Standard (14
Years)
    b.  47% of People Completed their Pre University Standard
(16 Years)
```

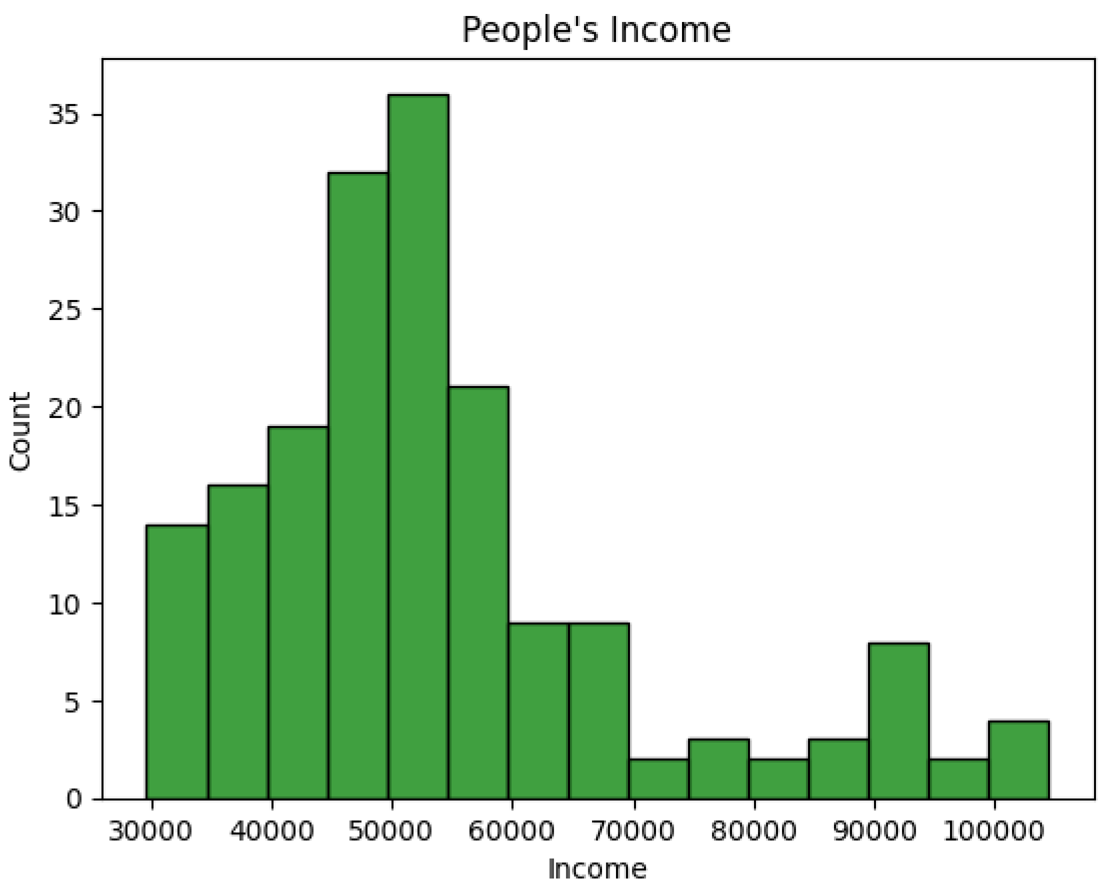
In [16]:
```
plt.hist(df['Fitness'],bins=7,color='brown')
plt.xlabel('Fitness')
plt.ylabel('Count')
plt.title('Fitness Levels')
plt.tight_layout()
plt.show()
```



Observations 💡

    53% Of People are Maintaining Average Fitness Level

In [17]:
```
sns.histplot(df['Income'],color='green')
plt.title("People's Income")
plt.show()
```

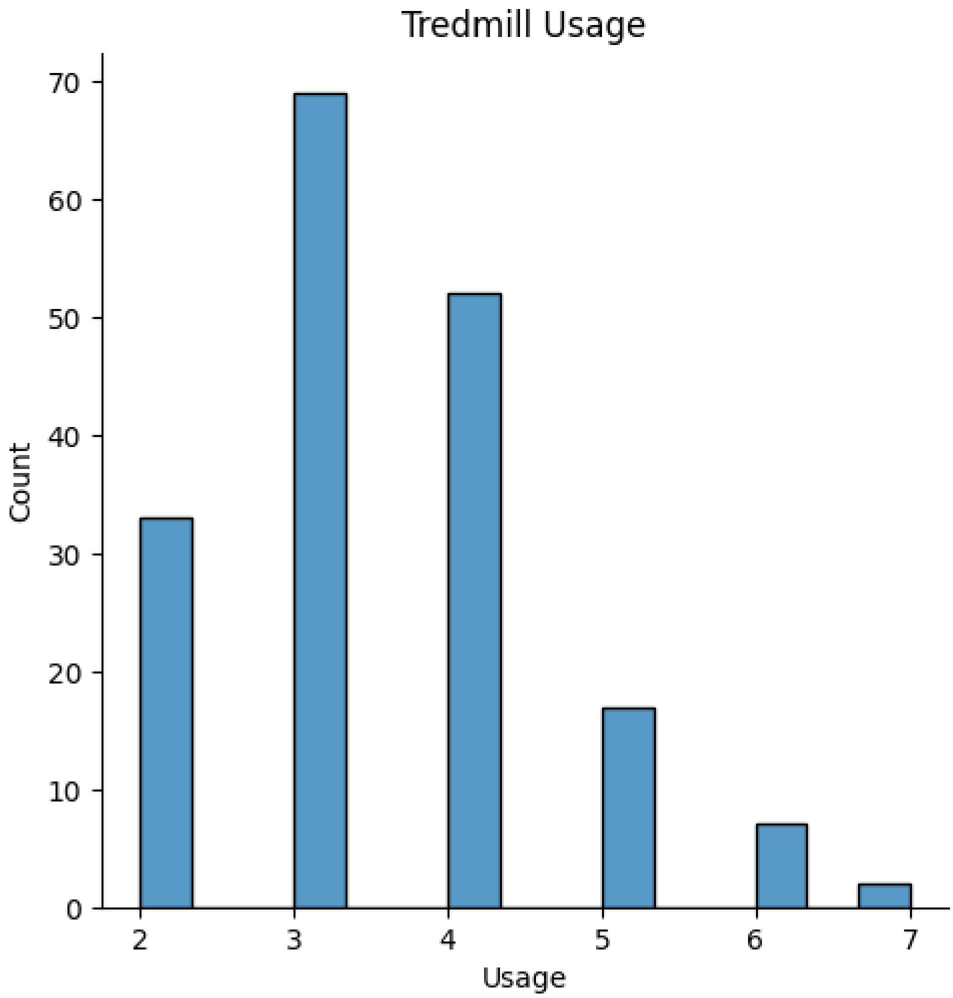


Observations 💡

```
76% Of People Have Annual Income less than 60K
```

In [18]:

```
sns.displot(df['Usage'])
plt.title('Tredmill Usage')
plt.show()
```



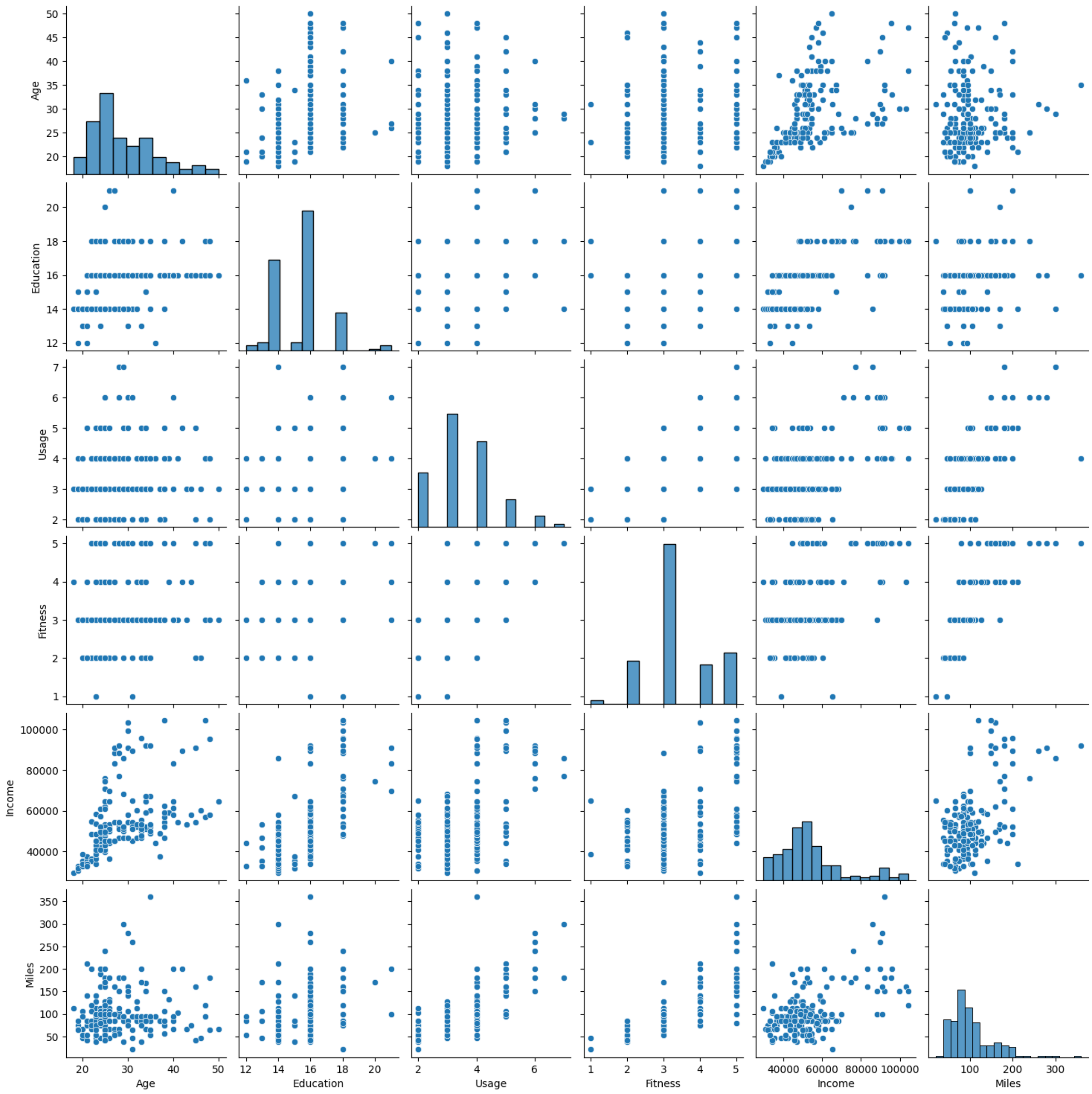
Observations 💡

    Maximum People Use tredmill Thrice in a Week

Bivariate Analysis

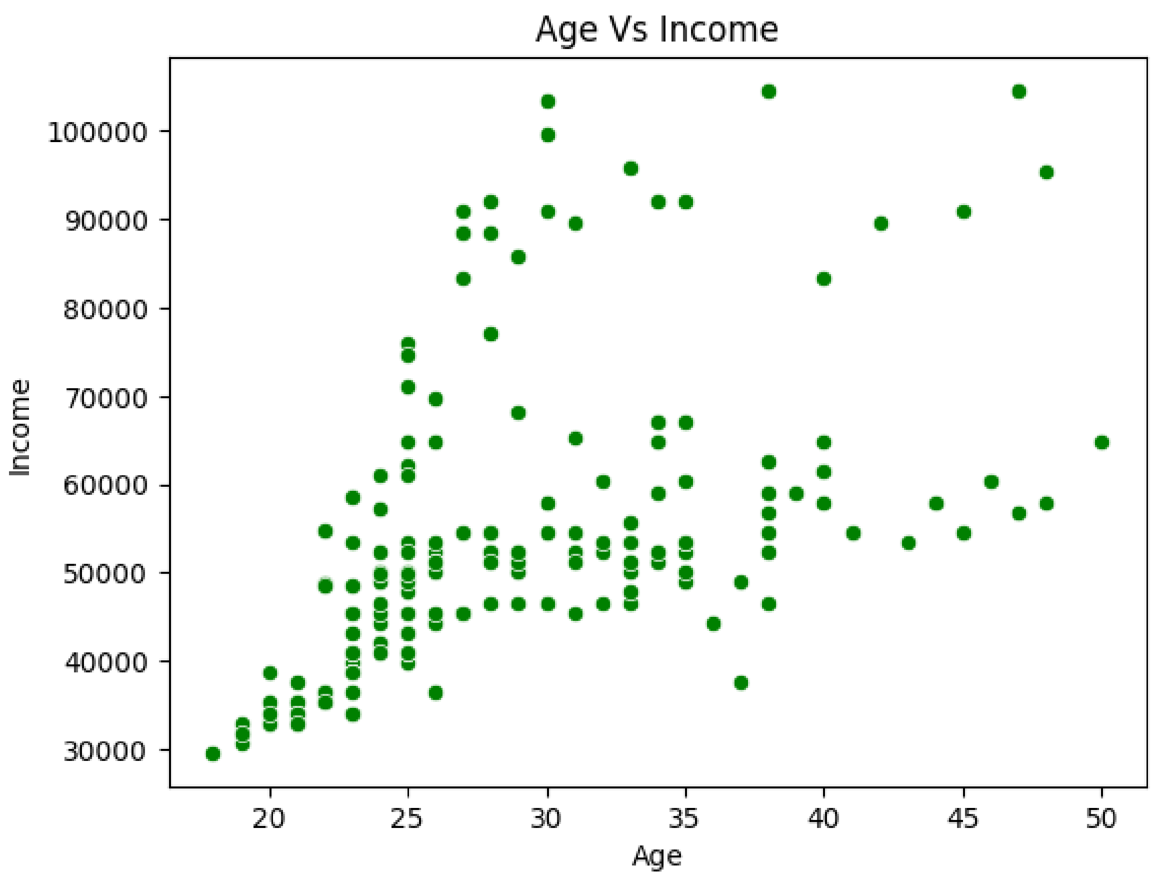
In [19]:
```
sns.pairplot(df)
```

Out[19]: <seaborn.axisgrid.PairGrid at 0x2a0397a93d0>



Age Vs Income

In [20]:
```
sns.scatterplot(data=df,x='Age',y='Income',c='g')
plt.title('Age Vs Income')
plt.show()
```

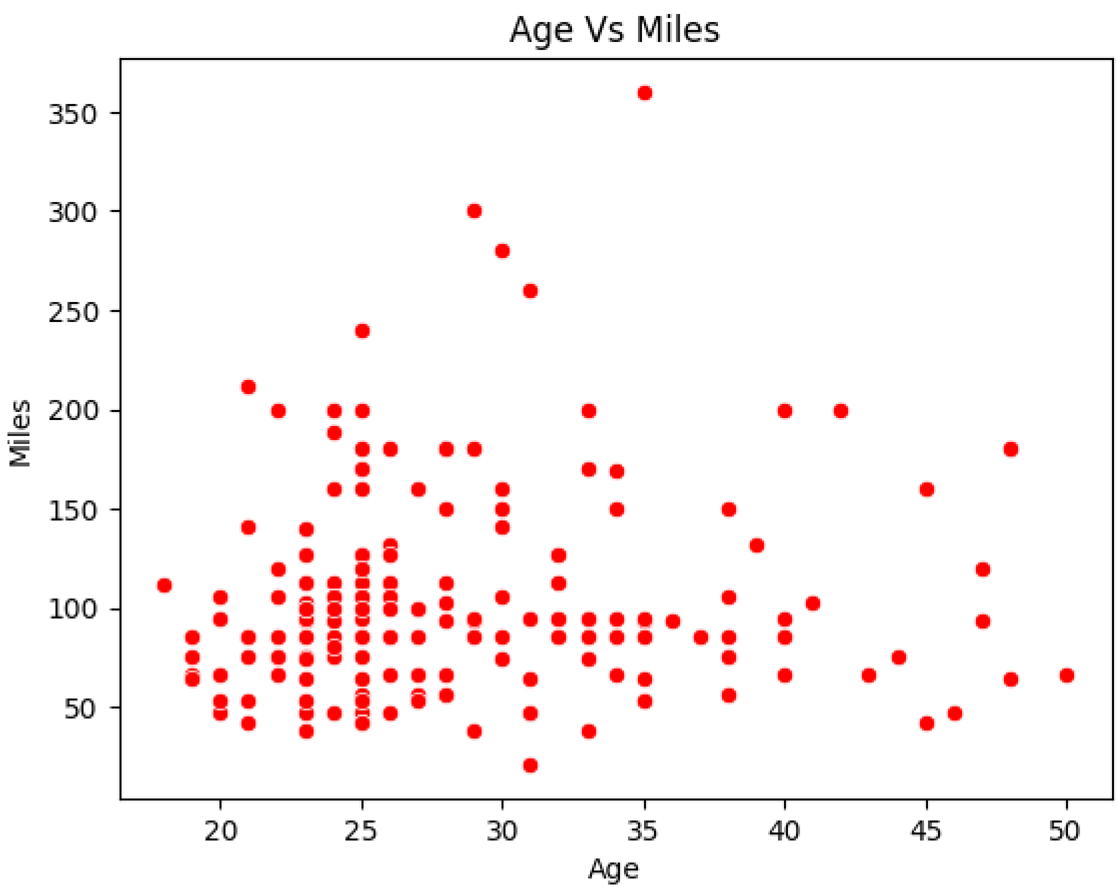


Observations 💡

    There is a Linear Relationship between Age & Income

Age Vs Miles

In [21]:
```
sns.scatterplot(data=df,x='Age',y='Miles',c='r')
plt.title('Age Vs Miles')
plt.show()
```



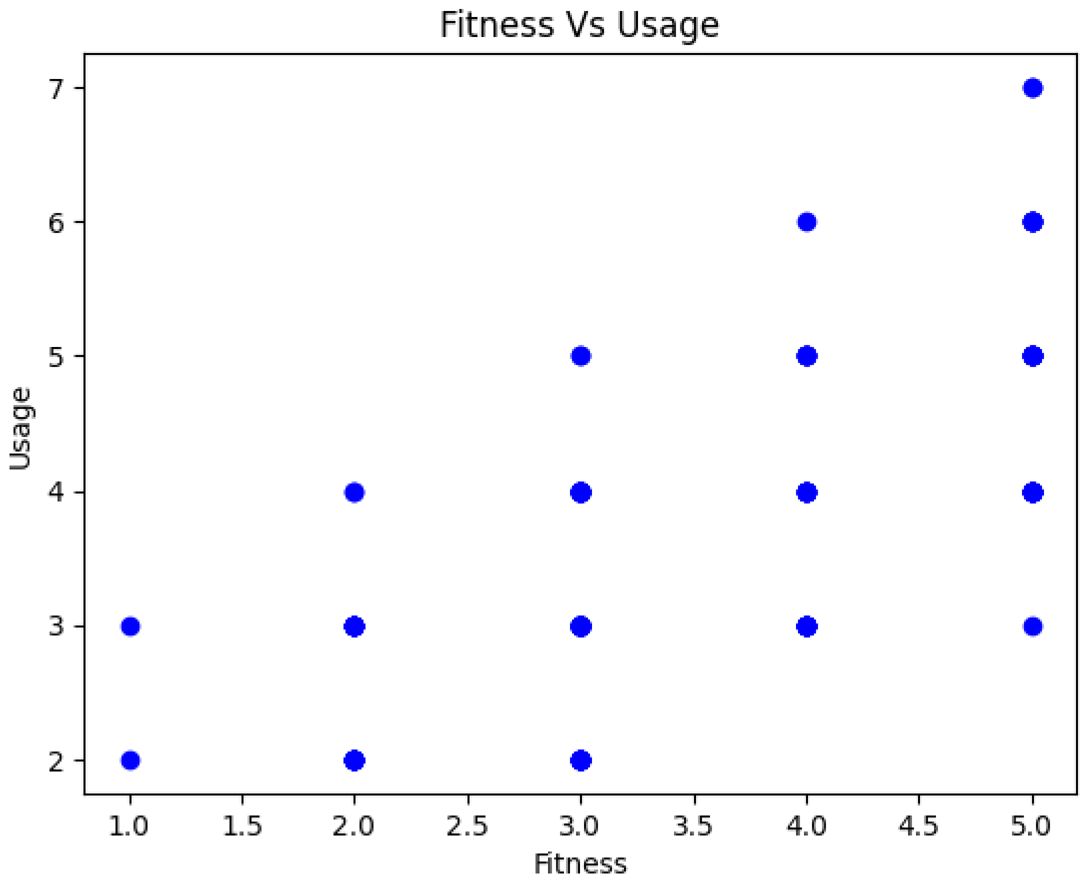
Observations 💡

```
upto 40 Age People are doing regerrsive workout miles above avg
i.e: 103
```

Fitness Vs Usage

In [22]:

```
plt.scatter(x=df['Fitness'],y=df['Usage'],c='b')
plt.xlabel('Fitness')
plt.ylabel('Usage')
plt.title('Fitness Vs Usage')
plt.show()
```



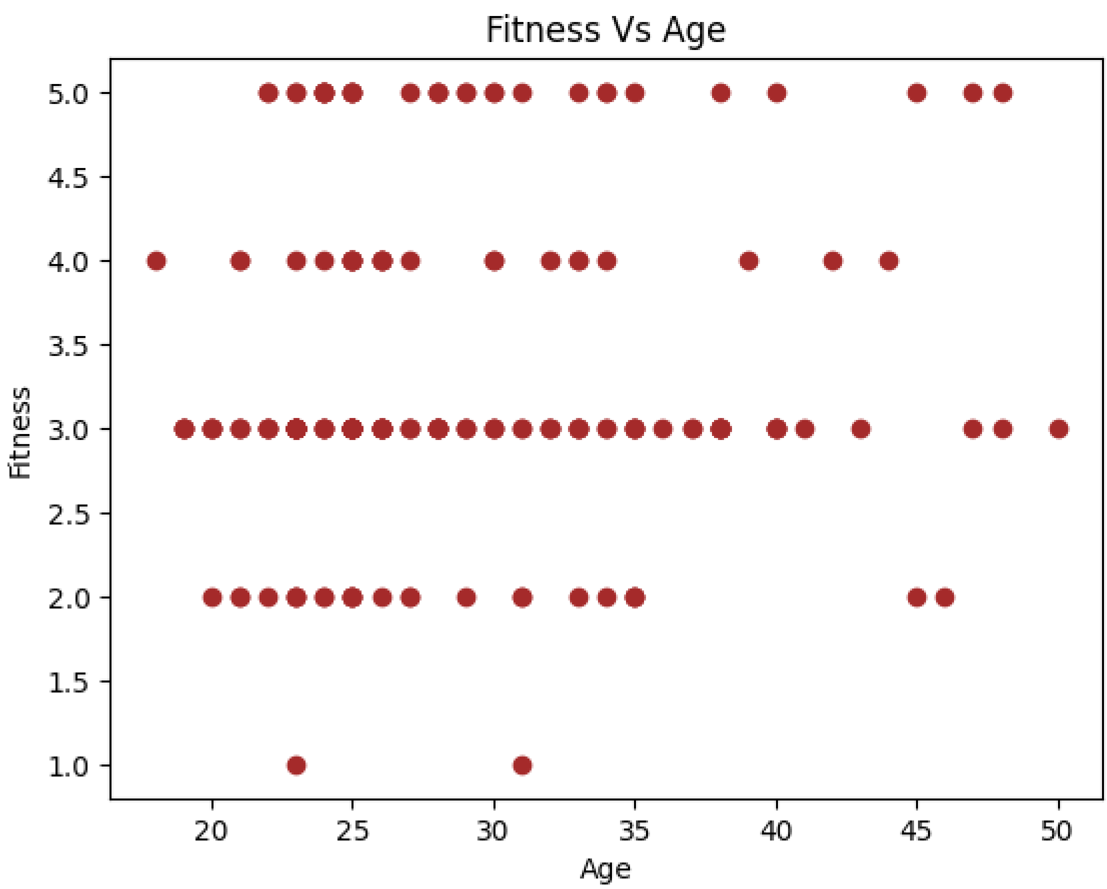
Observations 💡

    Whoever using Using Trendmill Everyday they have Excellent
    fitness levels

Fitness Vs Age

In [23]:

```
plt.scatter(y=df['Fitness'],x=df['Age'],c='brown')
plt.ylabel('Fitness')
plt.xlabel('Age')
plt.title('Fitness Vs Age')
plt.show()
```



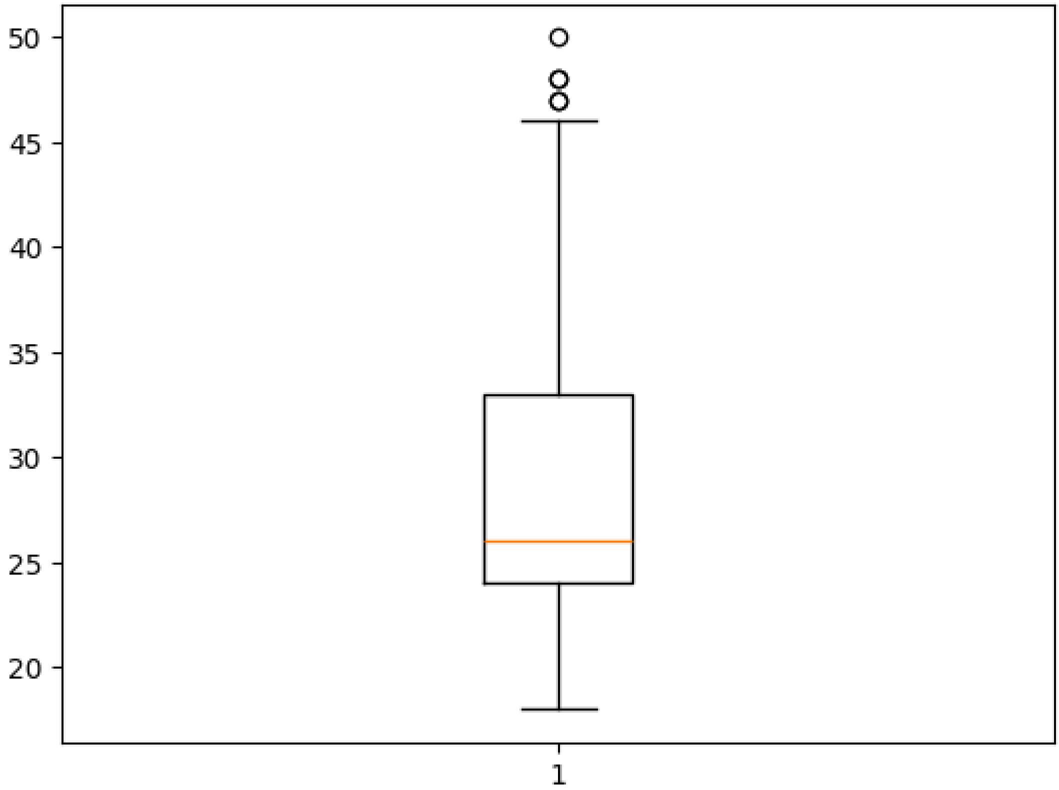
Observations 💡

    Upto 35 all people are caring about their fitness levels

Boxplots

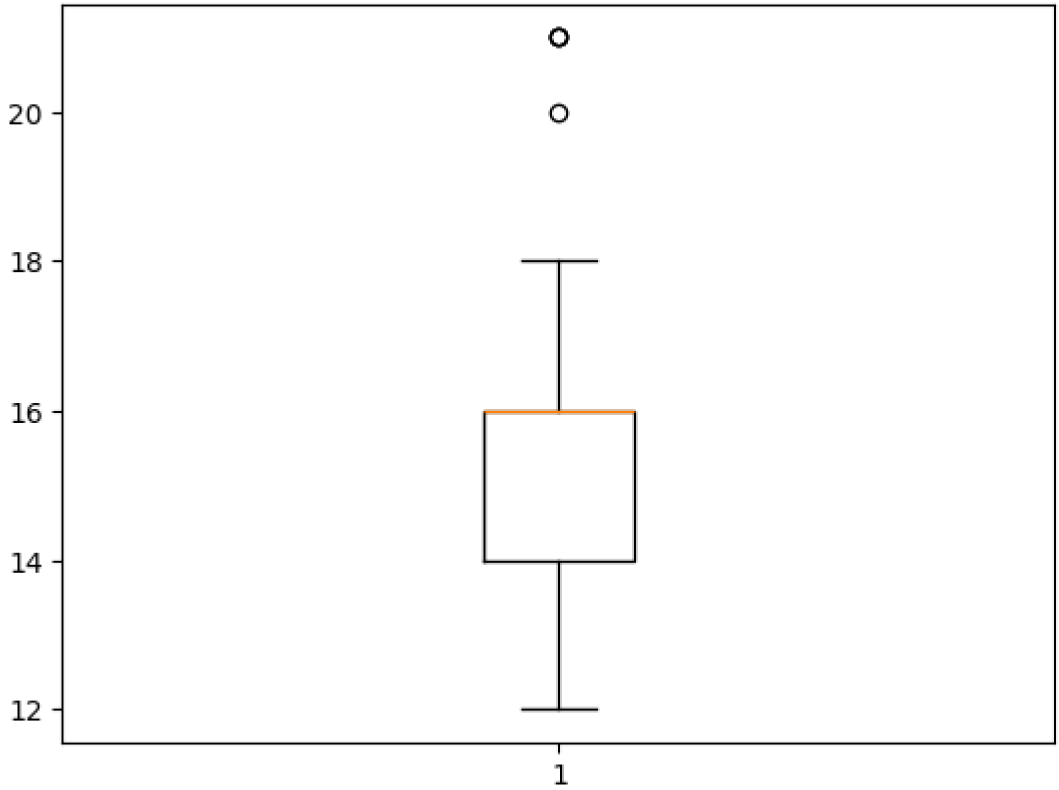
In [24]:
```
plt.boxplot(df['Age'])
data = df['Age']
q1 = np.percentile(data, 25)
q3 = np.percentile(data, 75)
iqr = q3 - q1
lower_bound = q1 - 1.5 * iqr
upper_bound = q3 + 1.5 * iqr
outliers = [x for x in data if x < lower_bound or x > upper_bound]
print(f"Outliers:", outliers)
plt.show()
```

```
Outliers: [47, 50, 48, 47, 48]
```



In [25]:
```
plt.boxplot(df['Education'])
data = df['Education']
q1 = np.percentile(data, 25)
q3 = np.percentile(data, 75)
iqr = q3 - q1
lower_bound = q1 - 1.5 * iqr
upper_bound = q3 + 1.5 * iqr
outliers = [x for x in data if x < lower_bound or x > upper_bound]
print(f"Outliers:", outliers)
plt.show()
```

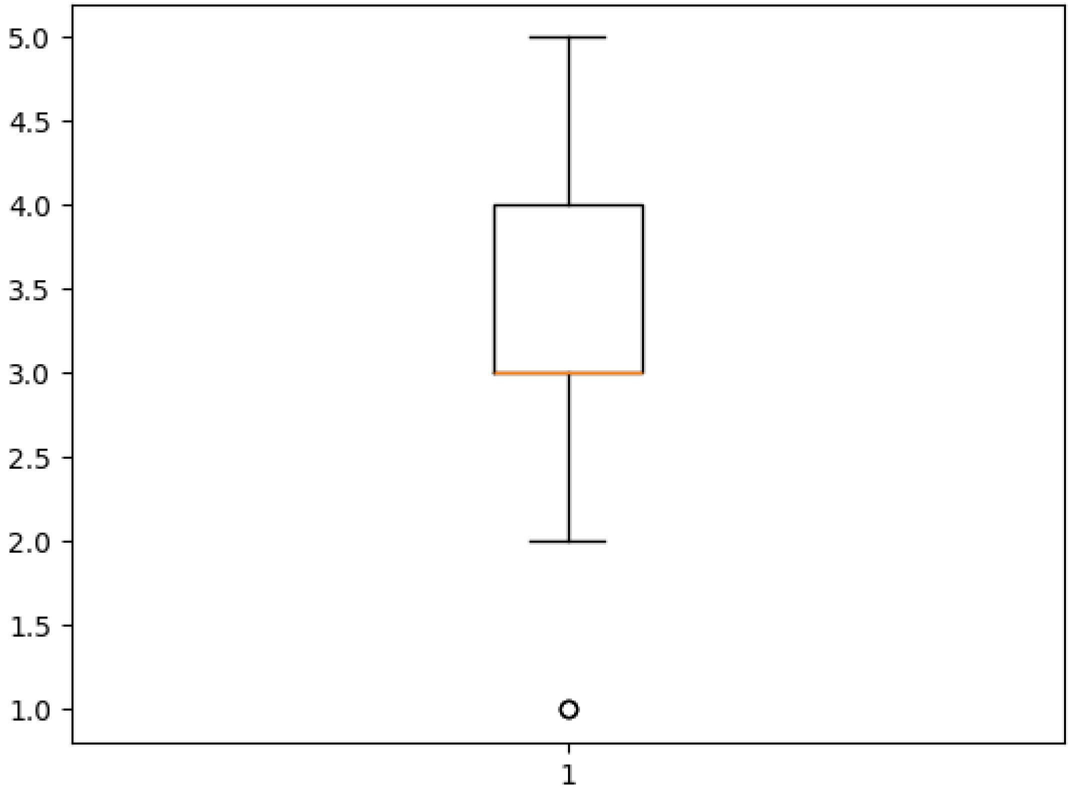
```
Outliers: [20, 21, 21, 21]
```



In [26]:

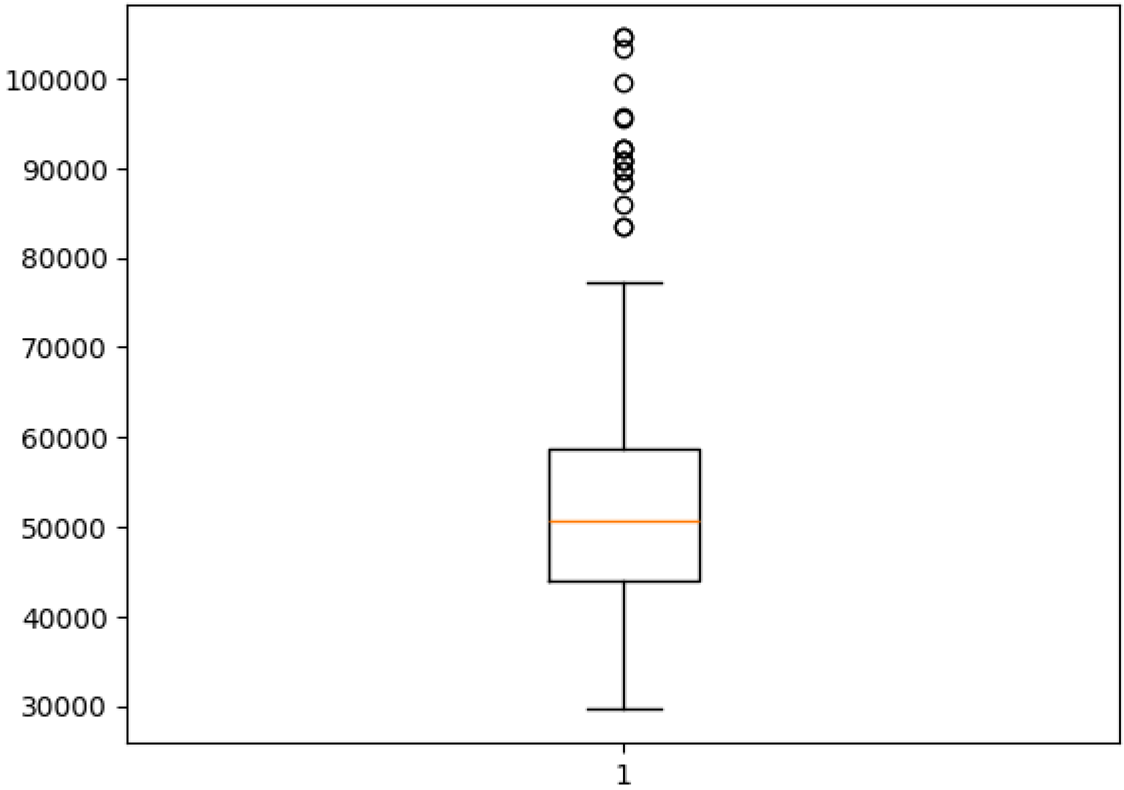
```
plt.boxplot(df['Fitness'])
data = df['Fitness']
q1 = np.percentile(data, 25)
q3 = np.percentile(data, 75)
iqr = q3 - q1
lower_bound = q1 - 1.5 * iqr
upper_bound = q3 + 1.5 * iqr
outliers = [x for x in data if x < lower_bound or x > upper_bound]
print(f"Outliers:", outliers)
plt.show()
```

```
Outliers: [1, 1]
```



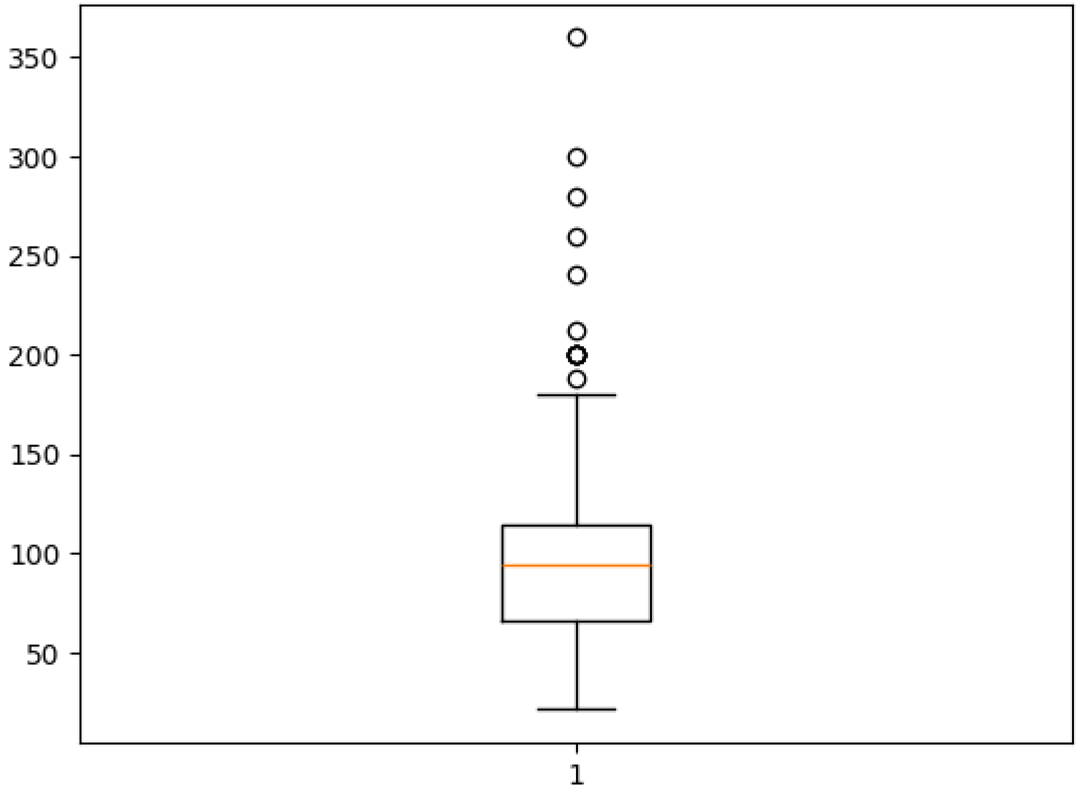
In [27]:
```
plt.boxplot(df['Income'])
data = df['Income']
q1 = np.percentile(data, 25)
q3 = np.percentile(data, 75)
iqr = q3 - q1
lower_bound = q1 - 1.5 * iqr
upper_bound = q3 + 1.5 * iqr
outliers = [x for x in data if x < lower_bound or x > upper_bound]
print(f"Outliers:", outliers)
plt.show()
```

```
Outliers: [83416, 88396, 90886, 92131, 88396, 85906, 90886, 103336, 99601, 89641,
95866, 92131, 92131, 104581, 83416, 89641, 90886, 104581, 95508]
```



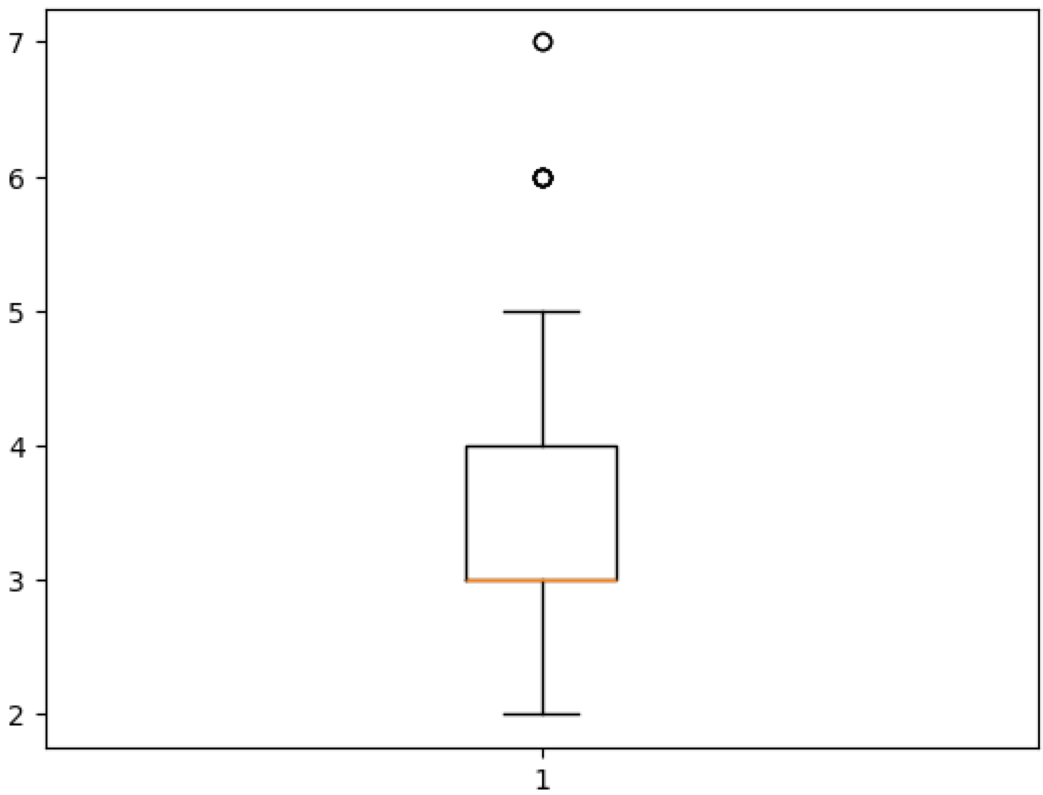
In [28]:
```
plt.boxplot(df['Miles'])
data = df['Miles']
q1 = np.percentile(data, 25)
q3 = np.percentile(data, 75)
iqr = q3 - q1
lower_bound = q1 - 1.5 * iqr
upper_bound = q3 + 1.5 * iqr
outliers = [x for x in data if x < lower_bound or x > upper_bound]
print(f"Outliers:", outliers)
plt.show()
```

```
Outliers: [188, 212, 200, 200, 200, 240, 300, 280, 260, 200, 360, 200, 200]
```



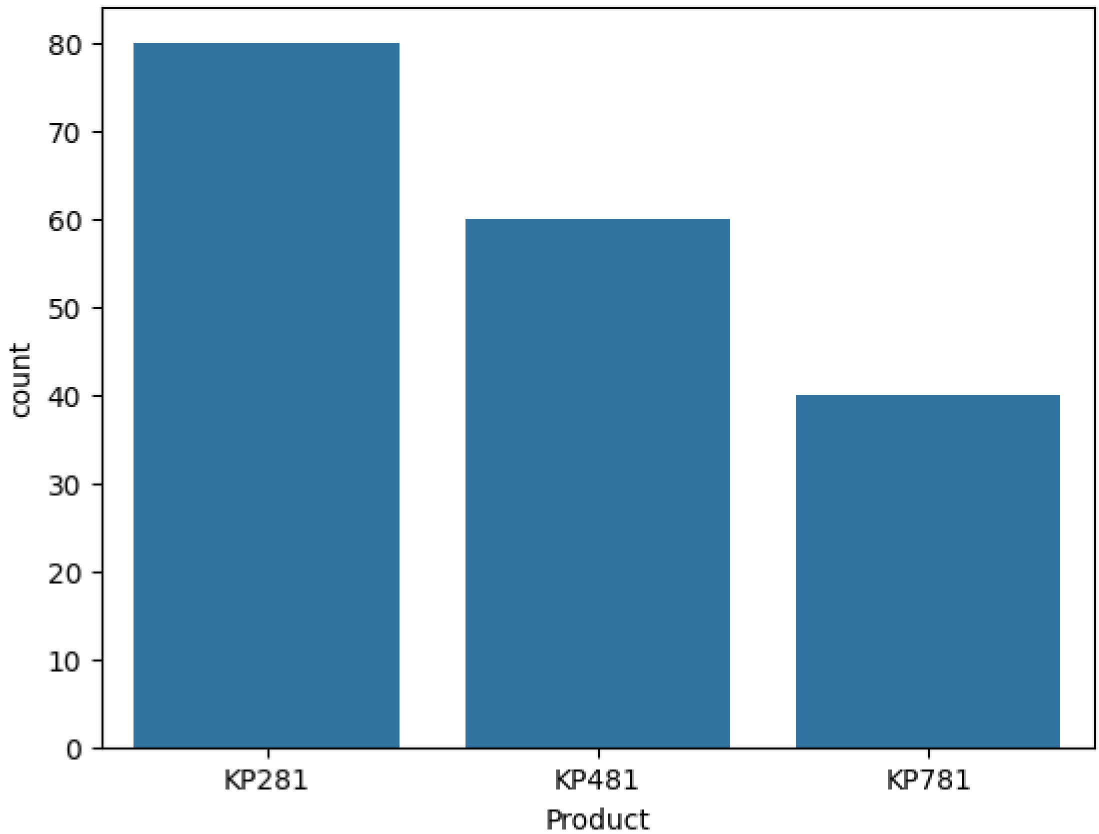
In [29]:
```
plt.boxplot(df['Usage'])
data = df['Usage']
q1 = np.percentile(data, 25)
q3 = np.percentile(data, 75)
iqr = q3 - q1
lower_bound = q1 - 1.5 * iqr
upper_bound = q3 + 1.5 * iqr
outliers = [x for x in data if x < lower_bound or x > upper_bound]
print(f"Outliers:", outliers)
plt.show()
```

```
Outliers: [6, 6, 6, 7, 6, 7, 6, 6, 6]
```



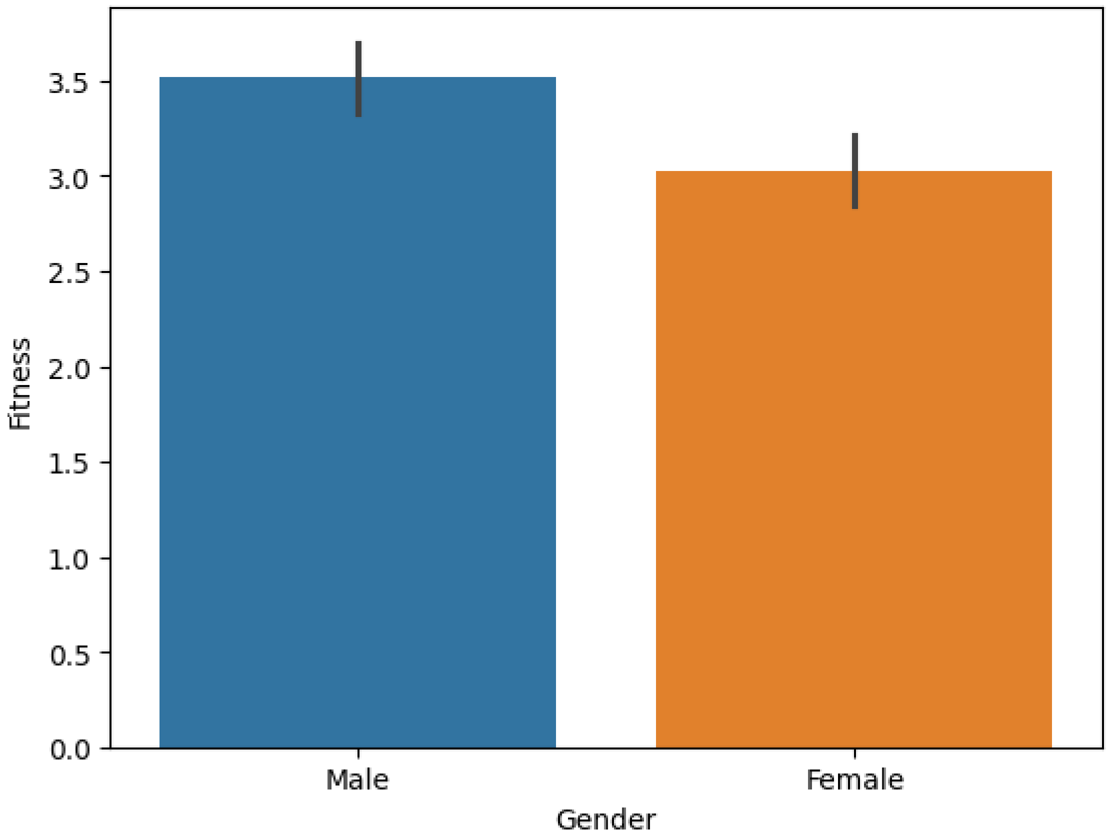
In [30]:
```
sns.countplot(df,x='Product')
```

Out[30]: <Axes: xlabel='Product', ylabel='count'>



In [31]:
```
sns.barplot(df,x='Gender',y='Fitness',hue="Gender")
```

Out[31]: <Axes: xlabel='Gender', ylabel='Fitness'>



Probability

Changing to Categorical Values for CrossTab

Observation 💡

```
Product, Gender, MaritalStatus are already in Categorical Form,
Rest of the columns needs to change to Categorica
```

In [32]:
```
df['Age'].unique()
```

Out[32]:
```
array([18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35, 36, 37, 38, 39, 40, 41, 43, 44, 46, 47, 50, 45, 48, 42],
      dtype=int64)
```

In [33]:
```
age_bins = [0, 28, 36,50]
age_labels = ['Youth', 'Middle-aged', 'Experienced']
df['Age'] = pd.cut(df['Age'], bins=age_bins, labels=age_labels)
df.head()
```

Out[33]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	KP281	Youth	Male	14	Single	3	4	29562	112
1	KP281	Youth	Male	15	Single	2	3	31836	75
2	KP281	Youth	Female	14	Partnered	4	3	30699	66
3	KP281	Youth	Male	12	Single	3	3	32973	85
4	KP281	Youth	Male	13	Partnered	4	2	35247	47

In [34]:
```
df['Age'].unique()
```

Out[34]:
```
['Youth', 'Middle-aged', 'Experienced']
Categories (3, object): ['Youth' < 'Middle-aged' < 'Experienced']
```

In [35]:
```
df['Education'].unique()
```

Out[35]:
```
array([14, 15, 12, 13, 16, 18, 20, 21], dtype=int64)
```

In [36]:
```
edu_bins = [0, 14, 16,21]
edu_labels = ['High School', 'Pre University', 'Graduate']
df['Education'] = pd.cut(df['Education'], bins=edu_bins, labels=edu_labels)
df.head()
```

Out[36]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	KP281	Youth	Male	High School	Single	3	4	29562	112
1	KP281	Youth	Male	Pre University	Single	2	3	31836	75
2	KP281	Youth	Female	High School	Partnered	4	3	30699	66
3	KP281	Youth	Male	High School	Single	3	3	32973	85
4	KP281	Youth	Male	High School	Partnered	4	2	35247	47

In [37]:
```
df['Education'].unique()
```

Out[37]:
```
['High School', 'Pre University', 'Graduate']
Categories (3, object): ['High School' < 'Pre University' < 'Graduate']
```

In [38]:
```
df['Usage'].unique()
```

Out[38]:
```
array([3, 2, 4, 5, 6, 7], dtype=int64)
```

In [39]:
```
usage_bins = [0, 2,3,4,5,6,7]
usage_labels = ['2Days', 'Thrice', '4Days','5Days','6Days','Daily']
df['Usage'] = pd.cut(df['Usage'], bins=usage_bins, labels=usage_labels)
df.head()
```

Out[39]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	KP281	Youth	Male	High School	Single	Thrice	4	29562	112
1	KP281	Youth	Male	Pre University	Single	2Days	3	31836	75
2	KP281	Youth	Female	High School	Partnered	4Days	3	30699	66
3	KP281	Youth	Male	High School	Single	Thrice	3	32973	85
4	KP281	Youth	Male	High School	Partnered	4Days	2	35247	47

In [40]:

```
df['Usage'].unique()
```

Out[40]:

```
['Thrice', '2Days', '4Days', '5Days', '6Days', 'Daily']
Categories (6, object): ['2Days' < 'Thrice' < '4Days' < '5Days' < '6Days' < 'Da
ily']
```

In [41]:

```
df['Fitness'].unique()
```

Out[41]:

```
array([4, 3, 2, 1, 5], dtype=int64)
```

In [42]:

```
fit_bins = [0, 1,2,3,4,5]
fit_labels = ['Poor', 'Below Average', 'Average','Above Average','Excellent']
df['Fitness'] = pd.cut(df['Fitness'], bins=fit_bins, labels=fit_labels)
df.head()
```

Out[42]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	KP281	Youth	Male	High School	Single	Thrice	Above Average	29562	112
1	KP281	Youth	Male	Pre University	Single	2Days	Average	31836	75
2	KP281	Youth	Female	High School	Partnered	4Days	Average	30699	66
3	KP281	Youth	Male	High School	Single	Thrice	Average	32973	85
4	KP281	Youth	Male	High School	Partnered	4Days	Below Average	35247	47

In [43]:

```
df['Fitness'].unique()
```

Out[43]:

```
['Above Average', 'Average', 'Below Average', 'Poor', 'Excellent']
Categories (5, object): ['Poor' < 'Below Average' < 'Average' < 'Above Average'
< 'Excellent']
```

In [44]:

```
df['Income'].unique()
```

Out[44]:
```
array([ 29562,  31836,  30699,  32973,  35247,  37521,  36384,  38658,
         40932,  34110,  39795,  42069,  44343,  45480,  46617,  48891,
         53439,  43206,  52302,  51165,  50028,  54576,  68220,  55713,
         60261,  67083,  56850,  59124,  61398,  57987,  64809,  47754,
         65220,  62535,  48658,  54781,  48556,  58516,  53536,  61006,
         57271,  52291,  49801,  62251,  64741,  70966,  75946,  74701,
         69721,  83416,  88396,  90886,  92131,  77191,  52290,  85906,
        103336,  99601,  89641,  95866, 104581,  95508], dtype=int64)
```

In [45]:
```
inc_bins = [0,35000,70000,104581]
inc_labels = ['Poor', 'Middle Class', 'Rich']
df['Income'] = pd.cut(df['Income'], bins=inc_bins, labels=inc_labels)
df.head()
```

Out[45]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	KP281	Youth	Male	High School	Single	Thrice	Above Average	Poor	112
1	KP281	Youth	Male	Pre University	Single	2Days	Average	Poor	75
2	KP281	Youth	Female	High School	Partnered	4Days	Average	Poor	66
3	KP281	Youth	Male	High School	Single	Thrice	Average	Poor	85
4	KP281	Youth	Male	High School	Partnered	4Days	Below Average	Middle Class	47

In [46]:
```
df['Miles'].unique()
```

Out[46]:
```
array([112,  75,  66,  85,  47, 141, 103,  94, 113,  38, 188,  56, 132,
       169,  64,  53, 106,  95, 212,  42, 127,  74, 170,  21, 120, 200,
       140, 100,  80, 160, 180, 240, 150, 300, 280, 260, 360], dtype=int64)
```

In [47]:
```
mil_bins = [0,100,200,360]
mil_labels = ['Casual', 'Regular', 'Advance']
df['Miles'] = pd.cut(df['Miles'], bins=mil_bins, labels=mil_labels)
df.head()
```

Out[47]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	KP281	Youth	Male	High School	Single	Thrice	Above Average	Poor	Regular
1	KP281	Youth	Male	Pre University	Single	2Days	Average	Poor	Casual
2	KP281	Youth	Female	High School	Partnered	4Days	Average	Poor	Casual
3	KP281	Youth	Male	High School	Single	Thrice	Average	Poor	Casual
4	KP281	Youth	Male	High School	Partnered	4Days	Below Average	Middle Class	Casual

In [48]:
```
df['Miles'].unique()
```

Out[48]:
```
['Regular', 'Casual', 'Advance']
Categories (3, object): ['Casual' < 'Regular' < 'Advance']
```

In [49]:
```
df.head()
```

Out[49]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	KP281	Youth	Male	High School	Single	Thrice	Above Average	Poor	Regular
1	KP281	Youth	Male	Pre University	Single	2Days	Average	Poor	Casual
2	KP281	Youth	Female	High School	Partnered	4Days	Average	Poor	Casual
3	KP281	Youth	Male	High School	Single	Thrice	Average	Poor	Casual
4	KP281	Youth	Male	High School	Partnered	4Days	Below Average	Middle Class	Casual

Product Vs Age

In [50]:
```
pd.crosstab(df['Product'],df['Age'],normalize=True,margins=True)
```

Out[50]:

Age	Youth	Middle-aged	Experienced	All
Product				
KP281	0.277778	0.094444	0.072222	0.444444
KP481	0.177778	0.111111	0.044444	0.333333
KP781	0.138889	0.050000	0.033333	0.222222
All	0.594444	0.255556	0.150000	1.000000

P(KP281 and Youth) = 0.27 P(KP281 and Middle-aged) = 0.94 P(KP281 and Experienced) = 0.07 P(KP481 and Youth) = 0.17 P(KP481 and Middle-aged) = 0.11 P(KP481 and Experienced) = 0.04 P(KP781 and Youth) = 0.13 P(KP781 and Middle-aged) = 0.05 P(KP781 and Experienced) = 0.03 P(KP281) = 0.44 P(KP481) = 0.33 P(KP781) = 0.22 P(Youth) = 0.59 P(Middle-aged) = 0.25

P(Experienced) = 0.33

In [75]:
```
pd.crosstab(df['Product'],df['Age'],normalize= 'columns',margins= True,
margins_name = 'Fraction_of_Product').round(2)
```

Out[75]:

Age	Youth	Middle-aged	Experienced	Fraction_of_Product
Product				
KP281	0.47	0.37	0.48	0.44
KP481	0.30	0.43	0.30	0.33
KP781	0.23	0.20	0.22	0.22

Conclusion 📒

```
1. 44% of The Customers Tend to buy KP281
2. 60% of Youth are going to buy Tredmill compared to other age
groups
3. All the age_groups prefer using KP281 are more compared to
other models.
```

Product Vs Gender

In [51]:

```python
pd.crosstab(df['Product'],df['Gender'],normalize=True,margins=True)
```

Out[51]:

Gender	Female	Male	All
Product			
KP281	0.222222	0.222222	0.444444
KP481	0.161111	0.172222	0.333333
KP781	0.038889	0.183333	0.222222
All	0.422222	0.577778	1.000000

P(KP281 and Female) = 0.22 P(KP281 and Male) = 0.22 P(KP481 and Female) = 0.16 P(KP481 and Male) = 0.17 P(KP781 and Female) = 0.03 P(KP781 and Male) = 0.18 P(KP281) = 0.44 P(KP481) = 0.33 P(KP781) = 0.22 P(Female) = 0.42 P(Male) = 0.57

In [77]:

```python
pd.crosstab(df['Product'],df['Gender'],normalize= 'columns',margins= True,
margins_name = 'Fraction_of_Product').round(2)
```

Out[77]:

Gender	Female	Male	Fraction_of_Product
Product			
KP281	0.53	0.38	0.44
KP481	0.38	0.30	0.33
KP781	0.09	0.32	0.22

Conclusion 📒

```
1. Again 44% of The Customers Tend to buy KP281
2. KP781 Tredmill is bought by Female is 3% only so don't Market
to Females KP781 tredmillro
```

Product Vs Education

In [52]:
```python
pd.crosstab(df['Product'],df['Education'],normalize=True,margins=True)
```

Out[52]:

Education / Product	High School	Pre University	Graduate	All
KP281	0.194444	0.238889	0.011111	0.444444
KP481	0.144444	0.177778	0.011111	0.333333
KP781	0.011111	0.083333	0.127778	0.222222
All	0.350000	0.500000	0.150000	1.000000

P(KP281 and High School) = 0.19 P(KP281 and Pre University) = 0.23 P(KP281 and Graduate) = 0.01 P(KP481 and High School) = 0.14 P(KP481 and Pre University) = 0.17 P(KP481 and Graduate) = 0.01 P(KP781 and High School) = 0.01 P(KP781 and Pre University) = 0.08 P(KP781 and Graduate) = 0.12 P(KP281) = 0.44 P(KP481) = 0.33 P(KP781) = 0.22 P(High School) = 0.35 P(Pre University) = 0.50 P(Graduate) = 0.15

In [78]:
```python
pd.crosstab(df['Product'],df['Education'],normalize= 'columns',margins= True,
margins_name = 'Fraction_of_Product').round(2)
```

Out[78]:

Education / Product	High School	Pre University	Graduate	Fraction_of_Product
KP281	0.56	0.48	0.07	0.44
KP481	0.41	0.36	0.07	0.33
KP781	0.03	0.17	0.85	0.22

Conclusion 📒

```
1. Again 44% of The Customers Tend to buy KP281
2. 50% of Tredmills are going to bought by Pre University
Educated People
3. KP281 & KP481 Tredmills are going to bought are not bougth by
Graduate People
```

Product Vs MaritalStatus

In [53]:
```python
pd.crosstab(df['Product'],df['MaritalStatus'],normalize=True,margins=True)
```

Out[53]:

MaritalStatus	Partnered	Single	All
Product			
KP281	0.266667	0.177778	0.444444
KP481	0.200000	0.133333	0.333333
KP781	0.127778	0.094444	0.222222
All	0.594444	0.405556	1.000000

P(KP281 and Partnered) = 0.26 P(KP281 and Single) = 0.17 P(KP481 and Partnered) = 0.20 P(KP481 and Single) = 0.13 P(KP781 and Partnered) = 0.12 P(KP781 and Single) = 0.09 P(KP281) = 0.44 P(KP481) = 0.33 P(KP781) = 0.22 P(Partnered) = 0.59 P(Single) = 0.40

In [79]:

```
pd.crosstab(df['Product'],df['MaritalStatus'],normalize= 'columns',margins= True
margins_name = 'Fraction_of_Product').round(2)
```

Out[79]:

MaritalStatus	Partnered	Single	Fraction_of_Product
Product			
KP281	0.45	0.44	0.44
KP481	0.34	0.33	0.33
KP781	0.21	0.23	0.22

Conclusion 📒

```
1. Again 44% of The Customers Tend to buy KP281
2. Single People are Buying 9% of KP781 Tredmills this is
something we need to take care
3. Partner, Single Ratio is 60:40
```

Product Vs Usage

In [54]:

```
pd.crosstab(df['Product'],df['Usage'],normalize=True,margins=True)
```

Out[54]:

Usage	2Days	Thrice	4Days	5Days	6Days	Daily	All
Product							
KP281	0.105556	0.205556	0.122222	0.011111	0.000000	0.000000	0.444444
KP481	0.077778	0.172222	0.066667	0.016667	0.000000	0.000000	0.333333
KP781	0.000000	0.005556	0.100000	0.066667	0.038889	0.011111	0.222222
All	0.183333	0.383333	0.288889	0.094444	0.038889	0.011111	1.000000

P(KP281 and 2Days) = 0.10 P(KP281 and Thrice) = 0.20 P(KP281 and 4Days) = 0.12 P(KP281 and 5Days) = 0.01 P(KP281 and 6Days) = 0.00 P(KP281 and Daily) = 0.00 P(KP481 and 2Days) = 0.07 P(KP481 and Thrice) = 0.17 P(KP481 and 4Days) = 0.06 P(KP481 and 5Days) = 0.01 P(KP481 and 6Days) = 0.00 P(KP481 and Daily) = 0.00 P(KP781 and 2Days) = 0.00 P(KP781 and Thrice) = 0.00 P(KP781 and 4Days) = 0.10 P(KP781 and 5Days) = 0.06 P(KP781 and 6Days) = 0.03 P(KP781 and Daily) = 0.01 P(KP281) = 0.44 P(KP481) = 0.33 P(KP781) = 0.22 P(2Days) = 0.18 P(Thrice) = 0.38 P(4Days) = 0.28 P(5Days) = 0.09 P(6Days) = 0.03 P(Daily) = 0.01

In [80]:
```
pd.crosstab(df['Product'],df['Usage'],normalize= 'columns',margins= True,
margins_name = 'Fraction_of_Product').round(2)
```

Out[80]:

Usage Product	2Days	Thrice	4Days	5Days	6Days	Daily	Fraction_of_Product
KP281	0.58	0.54	0.42	0.12	0.0	0.0	0.44
KP481	0.42	0.45	0.23	0.18	0.0	0.0	0.33
KP781	0.00	0.01	0.35	0.71	1.0	1.0	0.22

Conclusion 📒

```
1. Again 44% of The Customers Tend to buy KP281
2. Who Bought KP281 those are going to Thrice in a week, 4 Days
in Week rest of them are Very Minimal
3. Who Bought KP481 those are going to Thrice in a week, rest of
them are Very Minimal
4. 50% of People use tredmill 2Days,Thrice in a week
```

Product Vs Fitness

In [81]:
```
pd.crosstab(df['Product'],df['Fitness'],normalize=True,margins=True)
```

Out[81]:

Fitness Product	Poor	Below Average	Average	Above Average	Excellent	All
KP281	0.005556	0.077778	0.300000	0.050000	0.011111	0.444444
KP481	0.005556	0.066667	0.216667	0.044444	0.000000	0.333333
KP781	0.000000	0.000000	0.022222	0.038889	0.161111	0.222222
All	0.011111	0.144444	0.538889	0.133333	0.172222	1.000000

P(KP281 and Poor) = 0.19 P(KP281 and Below Average) = 0.23 P(KP281 and Average) = 0.01 P(KP281 and Above Average) = 0.23 P(KP281 and Excellent) = 0.01 P(KP481 and Poor) = 0.19 P(KP481 and Below Average) = 0.23 P(KP481 and Average) = 0.01 P(KP481 and Above Average) = 0.23 P(KP481 and Excellent) = 0.01 P(KP781 and Poor) = 0.19 P(KP781 and Below Average) = 0.23 P(KP781 and Average) = 0.01 P(KP781 and Above

Average) = 0.23 P(KP781 and Excellent) = 0.01 P(KP281) = 0.44 P(KP481) = 0.33 P(KP781) = 0.22 P(Poor) = 0.19 P(Below Average) = 0.23 P(Average) = 0.01 P(Above Average) = 0.23 P(Excellent) = 0.01

In [82]:
```
pd.crosstab(df['Product'],df['Fitness'],normalize= 'columns',margins= True,
margins_name = 'Fraction_of_Product').round(2)
```

Out[82]:

Fitness Product	Poor	Below Average	Average	Above Average	Excellent	Fraction_of_Product
KP281	0.5	0.54	0.56	0.38	0.06	0.44
KP481	0.5	0.46	0.40	0.33	0.00	0.33
KP781	0.0	0.00	0.04	0.29	0.94	0.22

Conclusion 📒

```
1. Again 44% of The Customers Tend to buy KP281
2. 30% KP281 Users Maintains Average Fitness Levels
3. 21% KP481 Users Maintains Average Fitness Levels
4. 53% of People Maintains Averge Fitnes Levels Irrespective of
which product they are using
```

Product Vs Income

In [56]:
```
pd.crosstab(df['Product'],df['Income'],normalize=True,margins=True)
```

Out[56]:

Income Product	Poor	Middle Class	Rich	All
KP281	0.044444	0.400000	0.000000	0.444444
KP481	0.033333	0.300000	0.000000	0.333333
KP781	0.000000	0.094444	0.127778	0.222222
All	0.077778	0.794444	0.127778	1.000000

P(KP281 and Poor) = 0.04 P(KP281 and Middle Class) = 0.40 P(KP281 and Rich) = 0.00 P(KP481 and Poor) = 0.03 P(KP481 and Middle Class) = 0.30 P(KP481 and Rich) = 0.00 P(KP781 and Poor) = 0.0 P(KP781 and Middle Class) = 0.09 P(KP781 and Rich) = 0.12 P(KP281) = 0.44 P(KP481) = 0.33 P(KP781) = 0.22 P(Poor) = 0.07 P(Middle Class) = 0.79 P(Rich) = 0.12

In [83]:
```
pd.crosstab(df['Product'],df['Income'],normalize= 'columns',margins= True,
margins_name = 'Fraction_of_Product').round(2)
```

Out[83]:

Income	Poor	Middle Class	Rich	Fraction_of_Product
Product				
KP281	0.57	0.50	0.0	0.44
KP481	0.43	0.38	0.0	0.33
KP781	0.00	0.12	1.0	0.22

Conclusion 📒

```
1. Again 44% of The Customers Tend to buy KP281
2. 40% of KP281 Tredmill is bought by Middle Class People
3. 30% of KP481 Tredmill is bought by Middle Class People
4. Only 12% Rich People are buying KP781 so Promote KP781 to
Rich People
```

Product Vs Miles

In [59]:
```
pd.crosstab(df['Product'],df['Miles'],normalize=True,margins=True)
```

Out[59]:

Miles	Casual	Regular	Advance	All
Product				
KP281	0.344444	0.100000	0.000000	0.444444
KP481	0.244444	0.083333	0.005556	0.333333
KP781	0.044444	0.150000	0.027778	0.222222
All	0.633333	0.333333	0.033333	1.000000

P(KP281 and Casual) = 0.34 P(KP281 and Regular) = 0.10 P(KP281 and Advance) = 0.0 P(KP481 and Casual) = 0.24 P(KP481 and Regular) = 0.08 P(KP481 and Advance) = 0.00 P(KP781 and Casual) = 0.04 P(KP781 and Regular) = 0.15 P(KP781 and Advance) = 0.02 P(KP281) = 0.44 P(KP481) = 0.33 P(KP781) = 0.22 P(Casual) = 0.63 P(Regular) = 0.33 P(Advance) = 0.03

In [84]:
```
pd.crosstab(df['Product'],df['Miles'],normalize= 'columns',margins= True,
margins_name = 'Fraction_of_Product').round(2)
```

Out[84]:

Miles	Casual	Regular	Advance	Fraction_of_Product
Product				
KP281	0.54	0.30	0.00	0.44
KP481	0.39	0.25	0.17	0.33
KP781	0.07	0.45	0.83	0.22

Recommendations

Customer Portfolio For KP281:

```
1. 44% of Users Bought this product
2. 60% of Youth Bought this product
3. 50% of the Pre University People Bought this product
4. The Usage of People are mainly in Thrice,4 Days week and also
30% of this people maintain Above Average Fitness Levels
5. 40% of This Product Customers are Middle Class
```

Customer Portfolio For KP481:

```
1. 33% Of Customers are tend to buy this product
2. The People who are using this product Manintains 2Days in a
week
3. 21% of Users Manintains Average Fitness Levels
4. 30% of this Product users Belongs to Middle Class
```

Customer Portfolio For KP781:

```
1. Customer Base for this product is only 20%
2. Only 3% Female People Bought this product
3. Only 9% of Single buy this product
4. Only 12% of Rich People bought this Product
```