Aerofit - Descriptive Statistics & Probability

Aerofit is a leading brand in the field of fitness equipment. Aerofit provides a product range including machines such as treadmills, exercise bikes, gym equipment, and fitness accessories to cater to the needs of all categories of people.

In [1]:
```
!gdown 1BORrVG7jVJltiRRVNEija6e9sIpG0GBi
```

```
Downloading...
From: https://drive.google.com/uc?id=1BORrVG7jVJltiRRVNEija6e9sIpG0GBi
To: /content/aerofit_treadmill.txt

  0% 0.00/7.28k [00:00<?, ?B/s]
100% 7.28k/7.28k [00:00<00:00, 2.74MB/s]
```

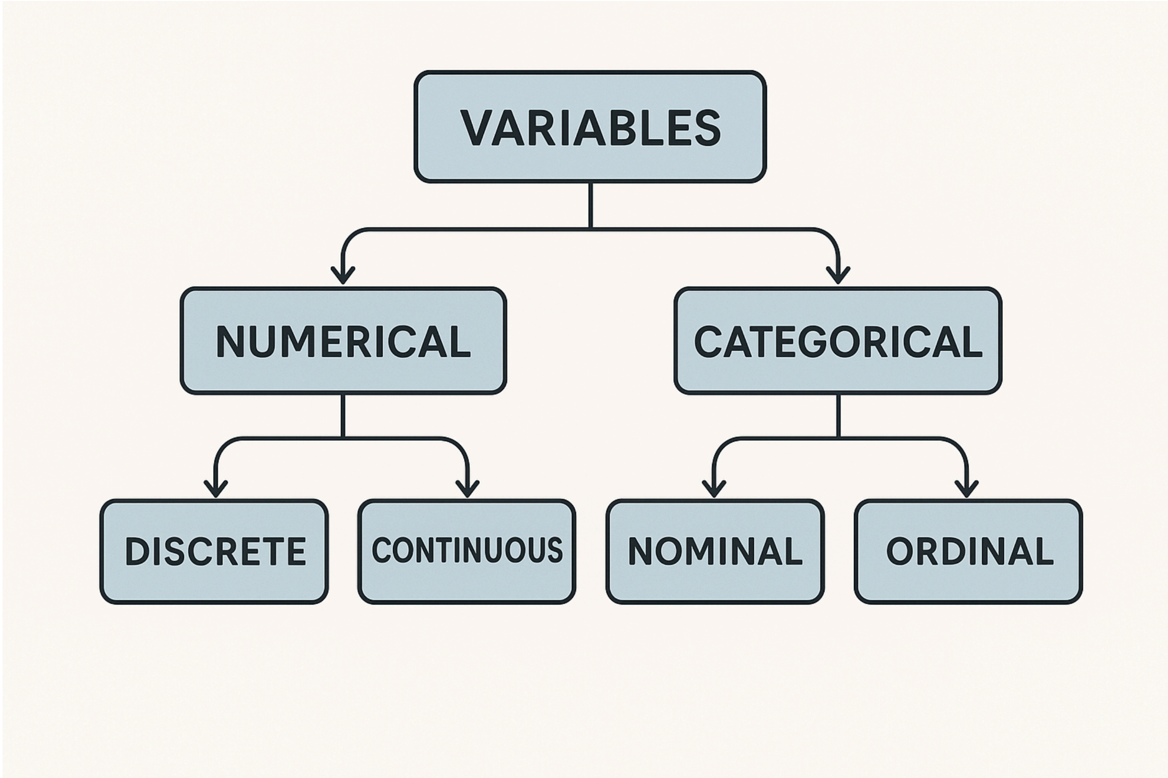
In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

data = pd.read_csv('aerofit_treadmill.txt')
data.head()
```

Out[2]:

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

Observation on variables & datatypes

Flow chart of variables



Columns: Miles , Income

Variable Type: Numerical → Continuous

Description:

Miles - Represents the distance covered, can take any real number value.

Income - Represents the value within a range and it's measurable.

Columns: Age , Education , Usage , Fitness

Variable Type: Numerical → Discrete

Description:

Age - The array contains whole number values, as represented in the dataset. However, conceptually, age is a continuous variable since it can take any value within a range (e.g., 20.5 years).

Usage - Represents average number of times per week , the customer used.

Education - Represents the person educated in years , can take in whole number.

Fitness - Represents the self-rated fitness on a 1-to-5 scale , can't take infinite values within a range.

Columns: Product , Gender , MaritalStatus

Variable Type: Categorical → Nominal

Product - Model numbers of trendmill , have no ranking in between them.

Gender - Represents the individual who purchased the trendmill.

MaritalStatus - Represents the relationship status of individual.

In [3]:
```
data.dtypes
```

Out[3]:

	0
Product	object
Age	int64
Gender	object
Education	int64
MaritalStatus	object
Usage	int64
Fitness	int64
Income	int64
Miles	int64

dtype: object

In [4]:
```
data.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 [5]:
```
print('Number of rows:',data.shape[0])
print('Number of columns:',data.shape[1])
```

```
Number of rows: 180
Number of columns: 9
```

1.Analysing the basic metrics and observation of Aerofit data & Problem Statement

In [6]:
```
# including all columns in a Dataframe description
data.describe(include='all')
```

Out[6]:

	Product	Age	Gender	Education	MaritalStatus	Usage
count	180	180.000000	180	180.000000	180	180.000000
unique	3	NaN	2	NaN	2	NaN
top	KP281	NaN	Male	NaN	Partnered	NaN
freq	80	NaN	104	NaN	107	NaN
mean	NaN	28.788889	NaN	15.572222	NaN	3.455556
std	NaN	6.943498	NaN	1.617055	NaN	1.084797
min	NaN	18.000000	NaN	12.000000	NaN	2.000000
25%	NaN	24.000000	NaN	14.000000	NaN	3.000000
50%	NaN	26.000000	NaN	16.000000	NaN	3.000000
75%	NaN	33.000000	NaN	16.000000	NaN	4.000000
max	NaN	50.000000	NaN	21.000000	NaN	7.000000

In [7]:
```
# including categorical columns in a Dataframe description
data.describe(include='object')
```

Out[7]:

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

In [8]:
```
# including numerical columns in a Dataframe description
data.describe(include=[np.number])
```

Out[8]:

	Age	Education	Usage	Fitness	Income	Mile
count	180.000000	180.000000	180.000000	180.000000	180.000000	180.00000
mean	28.788889	15.572222	3.455556	3.311111	53719.577778	103.19444
std	6.943498	1.617055	1.084797	0.958869	16506.684226	51.86360
min	18.000000	12.000000	2.000000	1.000000	29562.000000	21.00000
25%	24.000000	14.000000	3.000000	3.000000	44058.750000	66.00000
50%	26.000000	16.000000	3.000000	3.000000	50596.500000	94.00000
75%	33.000000	16.000000	4.000000	4.000000	58668.000000	114.75000
max	50.000000	21.000000	7.000000	5.000000	104581.000000	360.00000

In [9]:

```
# Analysing the nan values for each column
data.isna().sum()/len(data)*100
```

Out[9]:

	0
Product	0.0
Age	0.0
Gender	0.0
Education	0.0
MaritalStatus	0.0
Usage	0.0
Fitness	0.0
Income	0.0
Miles	0.0

dtype: float64

Conversions of columns

- Converting the categorical attributes columns to category
- Adding level of education - Better visualisation

In [10]:

```
data[['Product','Gender','MaritalStatus']] = data[['Product','Gender','Marital
```

In [11]:

```
data.loc[data['Education']==12, 'level of Education']='High School'
data.loc[(data['Education']==13) | (data['Education']==14) | (data['Education'
data.loc[(data['Education']==17) | (data['Education']==18) , 'level of Educati
data.loc[(data['Education']==20) | (data['Education']==21) ,'level of Educatio
```

Problem Statement:

As a Data Scientist at Aerofit , I have been tasked with the responsibility of analyzing the provided dataset to extract valuable insights and deliver actionable recommendations.

It could be:

- Defining the relation between the gender and fitness levels
- Analysing the probability of each product with respect to the variables.
- Creating the beautiful insights and business recommendations to grow business.

2.Non-Graphical Analysis

In [12]:
```
# Number of unique values in every column of dataset

for col in data.columns:
  print(col,':',data[col].nunique())
```

```
Product : 3
Age : 32
Gender : 2
Education : 8
MaritalStatus : 2
Usage : 6
Fitness : 5
Income : 62
Miles : 37
level of Education : 4
```

In [13]:
```
# Unique values in every column of dataset

for col in data.columns:
  print(col,':',data[col].unique())
```

```
Product : ['KP281', 'KP481', 'KP781']
Categories (3, object): ['KP281', 'KP481', 'KP781']
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]
Gender : ['Male', 'Female']
Categories (2, object): ['Female', 'Male']
Education : [14 15 12 13 16 18 20 21]
MaritalStatus : ['Single', 'Partnered']
Categories (2, object): ['Partnered', 'Single']
Usage : [3 2 4 5 6 7]
Fitness : [4 3 2 1 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]
Miles : [112  75  66  85  47 141 103  94 113  38 188  56 132 169  64  53 106  9
5
 212  42 127  74 170  21 120 200 140 100  80 160 180 240 150 300 280 260
 360]
level of Education : ['Bachelor' 'High School' 'Master' 'Doctorate/Ph.D']
```

In [14]:

```
# Observation of value counts in each column (first four columns)
for col in data.columns[0:4]:
  print(col,':',data[col].value_counts())
```

```
Product : Product
KP281    80
KP481    60
KP781    40
Name: count, dtype: int64
Age : Age
25    25
23    18
24    12
26    12
28     9
33     8
35     8
22     7
30     7
27     7
38     7
21     7
31     6
34     6
29     6
20     5
40     5
19     4
32     4
37     2
45     2
48     2
47     2
18     1
41     1
39     1
36     1
43     1
46     1
44     1
50     1
42     1
Name: count, dtype: int64
Gender : Gender
Male      104
Female     76
Name: count, dtype: int64
Education : Education
16    85
14    55
18    23
15     5
13     5
12     3
21     3
20     1
Name: count, dtype: int64
```

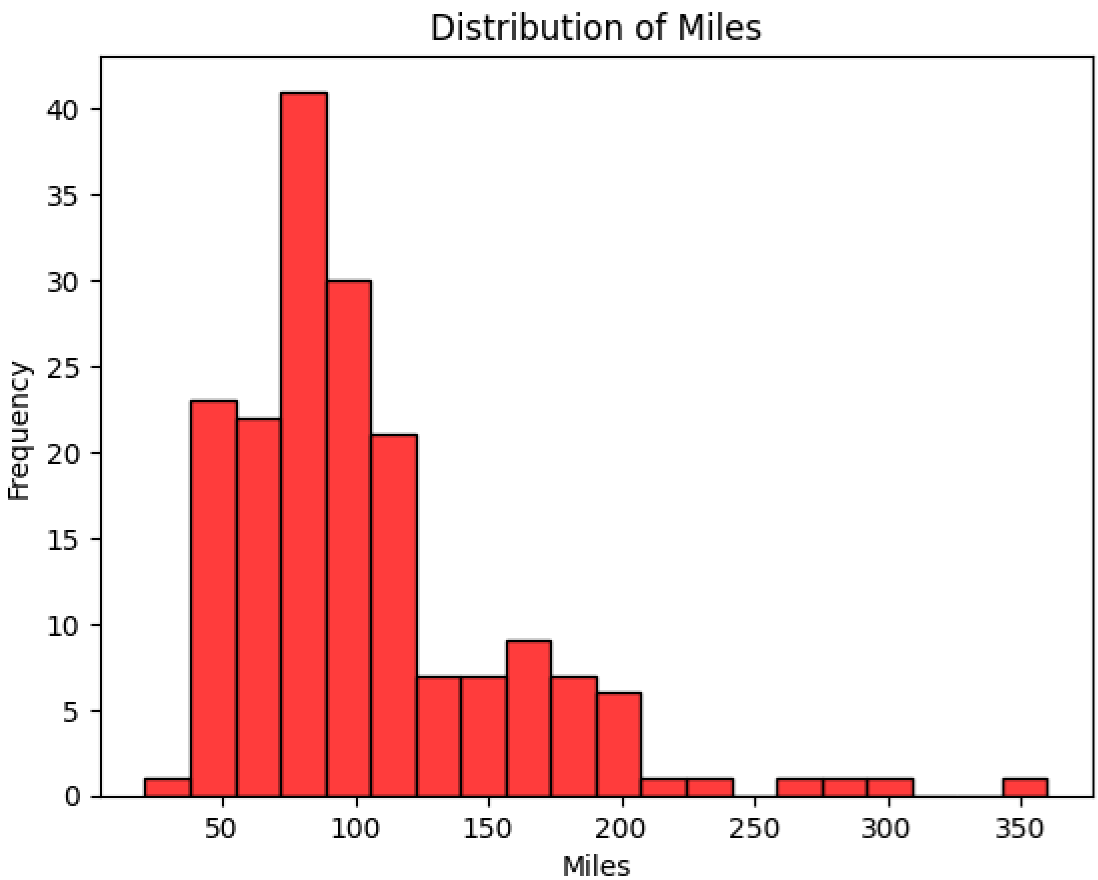
In [15]:
```
#Examine of duplicated values in dataset
data.duplicated().sum()
```

Out[15]: np.int64(0)

3.Visual Analysis - Univariate

Hist plot - Continuous variable(Miles , Income)

In [16]:
```
sns.histplot(data=data,x='Miles',color='Red')
plt.xlabel('Miles')
plt.ylabel('Frequency')
plt.title('Distribution of Miles')
plt.show()
```

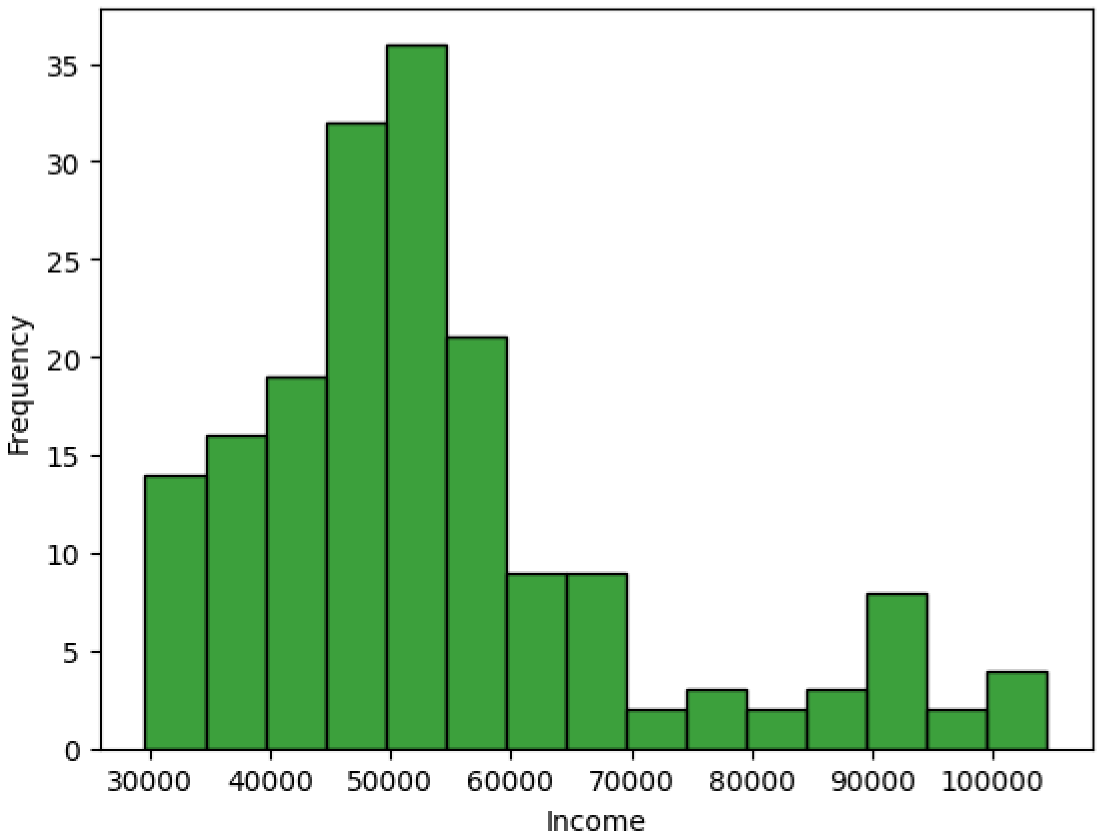


Observation :

- The histogram shows a **right-skewed distribution**, indicating that most individuals walk fewer miles.
- The distribution of weekly miles walked by individuals ranges from 21 to 360 miles.

- Within a range of 21 to 360 miles (minimum to maximum), out of 180 people, the highest number of individuals—approximately 27—walk about 85 miles per week.
- The highest frequency of walkers is in the 60–100 miles range.

In [17]:
```
sns.histplot(data=data,x='Income',color='Green')
plt.xlabel('Income')
plt.ylabel('Frequency')
plt.show()
```



Observation :

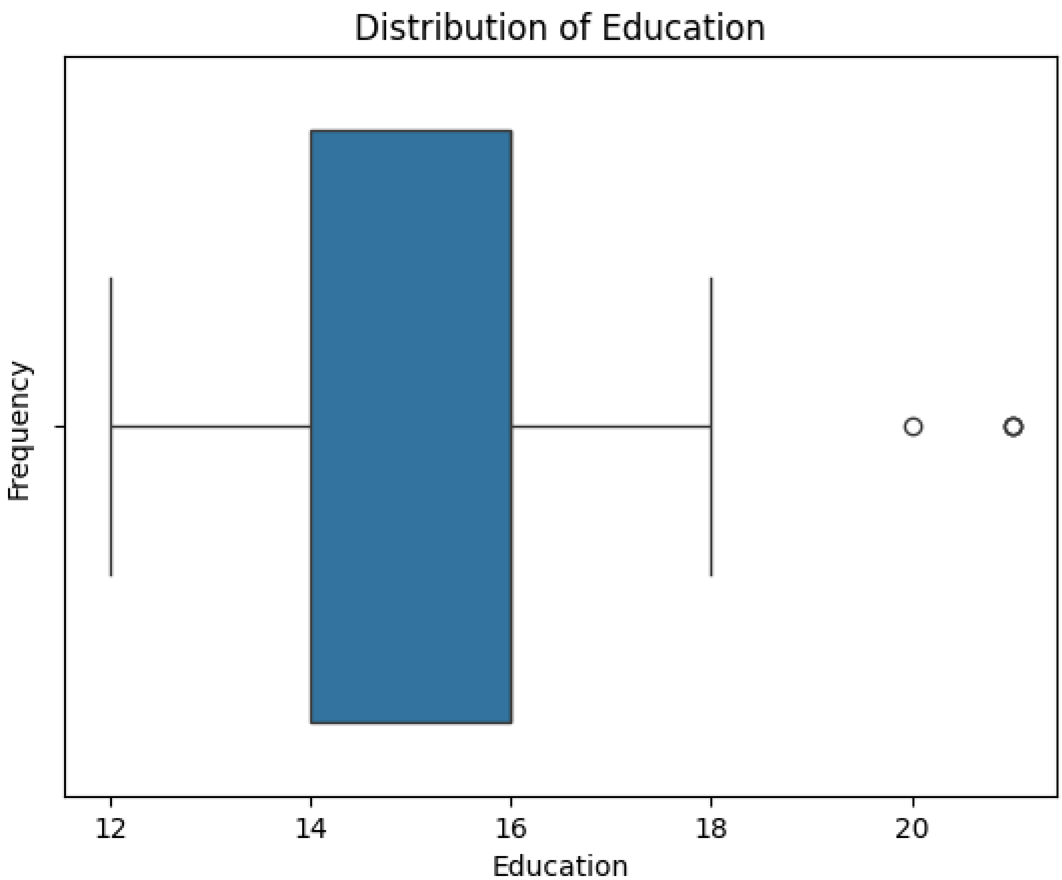
- The histogram shows a **right-skewed distribution**, indicating that most individuals earns huge income.
- The distribution of annual income by individuals ranges from 20,000 USD to 1,00,000 USD.
- The highest spike of income by individuals ranges between 45,000 USD - 55,000 USD
- Out of 180 people, the highest number of individuals — approximately

27 - earns 45,000 USD per annual.

Boxplot - Categorical variable(Product , Gender , MaritalStatus)

Based on data,we can't make a boxplot of a categorical variable.

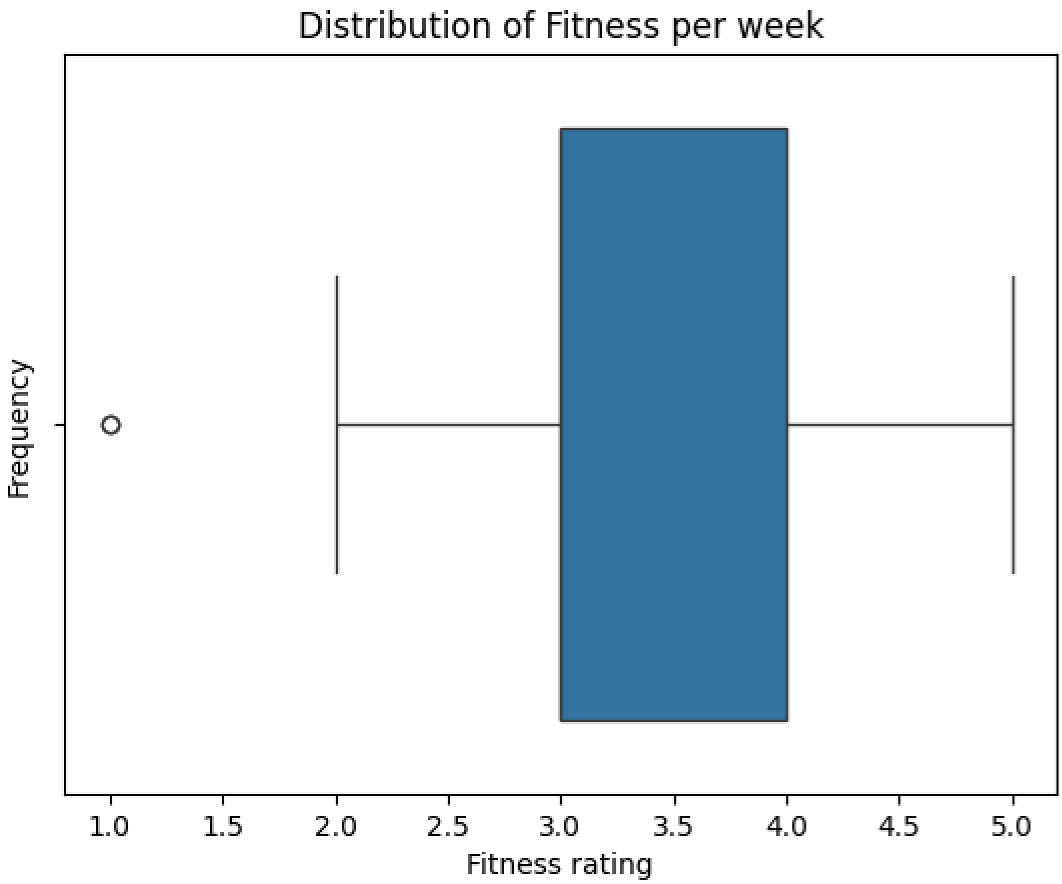
In [18]:
```
sns.boxplot(data=data,x='Education')
plt.xlabel('Education')
plt.ylabel('Frequency')
plt.title('Distribution of Education')
plt.show()
```



Observations :

- The distribution of education in the boxplot is right-skewed **distribution**, with a declining tail toward higher education levels (18–20 years).
- The distribution of education by individuals ranges from 12 to 20 years.
- Out of 180 people, the highest number of individuals — approximately 140 - pursuing their bachelor's education.

In [19]:
```
sns.boxplot(data=data,x='Fitness')
plt.xlabel('Fitness rating')
plt.ylabel('Frequency')
plt.title('Distribution of Fitness per week')
plt.show()
```



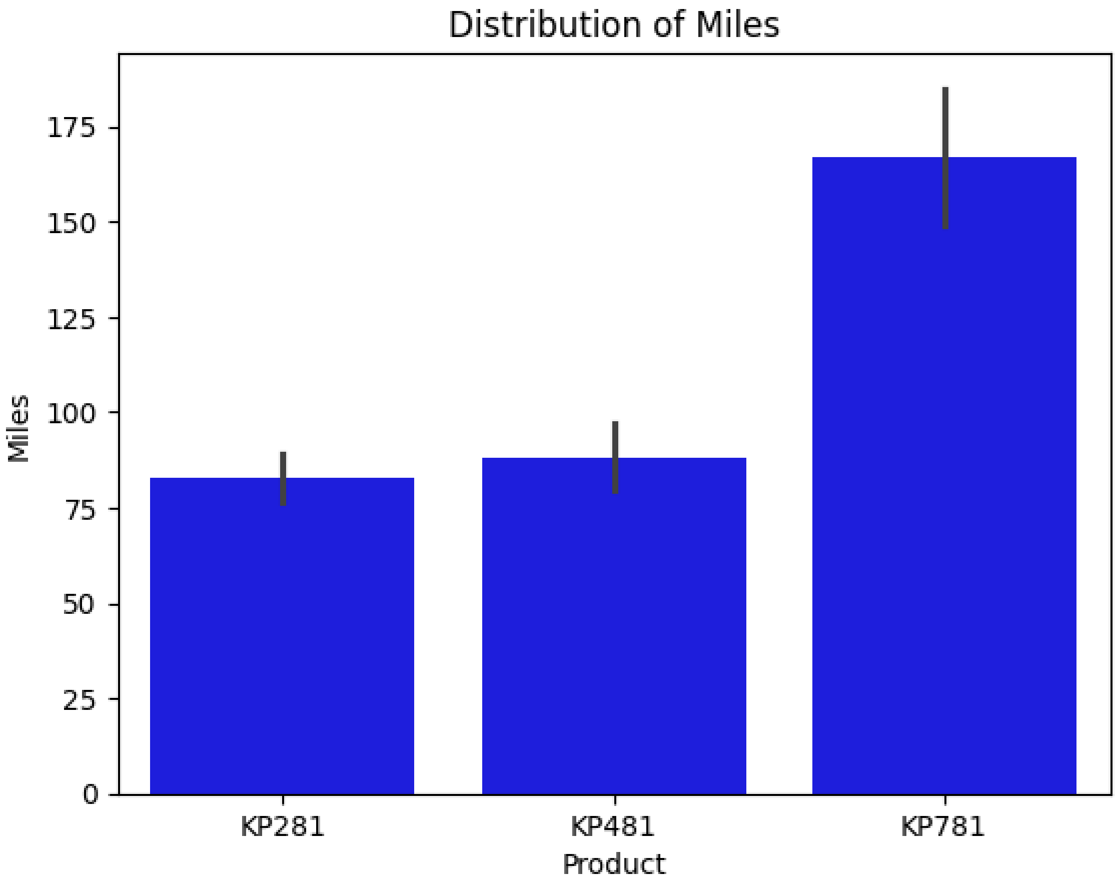
Observations :

- The boxplot shows a left-skewed distribution of fitness ratings, with a longer tail towards the lower end (ratings 1–2).
- Fitness ratings range from 1 (poor) to 5 (excellent).
- Out of 180 individuals, the majority — around 97 people — rated their fitness as 3, indicating an average level of physical activity per week.
- Only two individuals gave a low fitness rating, suggesting they are minimally or not physically active.

Barplot - Categorical values

In [20]:
```
sns.barplot(data=data,x='Product',y='Miles',color='blue')
plt.xlabel('Product')
```

```
plt.ylabel('Miles')
plt.title('Distribution of Miles')
plt.show()
```

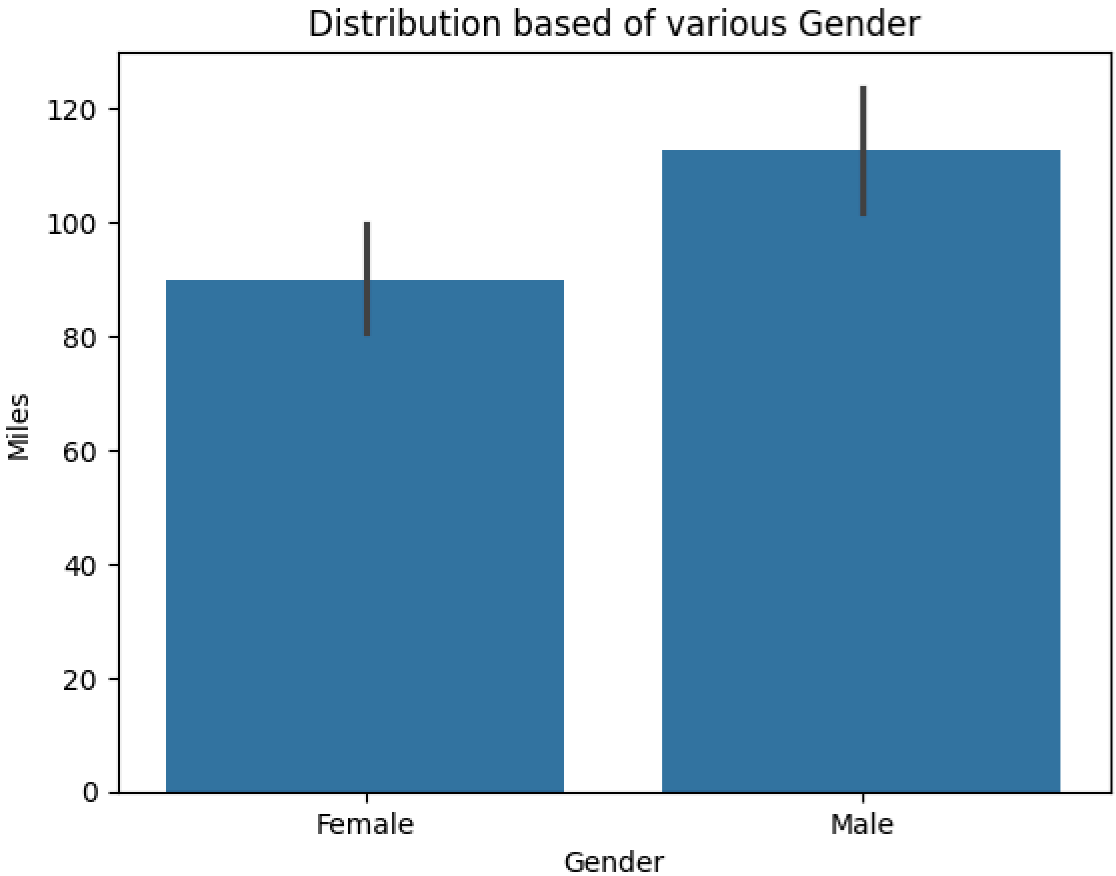


Observations:

- The barplot shows a distribution of various Product considering KP281, KP481 and KP781.
- KP781 not only has the highest average miles, but also the largest variability among the three products.
- KP781 is the most reliable choice.
- KP281: Around 75 miles
- KP781: Around 175 miles
- KP481: Around 100 miles

In [21]:
```
sns.barplot(data=data,x='Gender',y='Miles')
plt.xlabel('Gender')
plt.ylabel('Miles')
plt.title('Distribution based of various Gender')
```

```
plt.show()
```

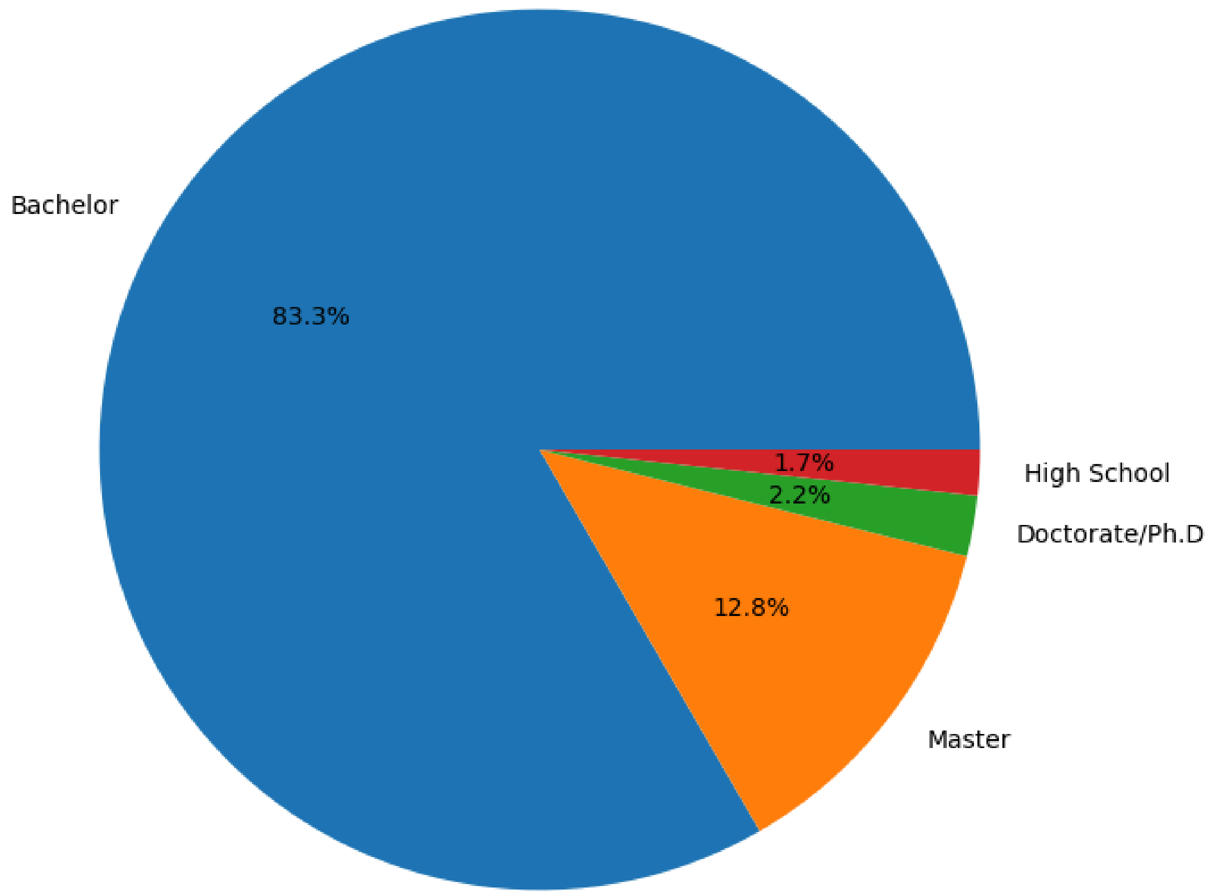


Observations :

- On average, males have covered more miles than females.
- This could imply higher physical activity or greater treadmill usage among males in your dataset.

Pie Chart - categorical values

In [22]:
```
plt.figure(figsize=(8,8))
plt.pie(data['level of Education'].value_counts(),labels=data['level of Educat
plt.show()
```

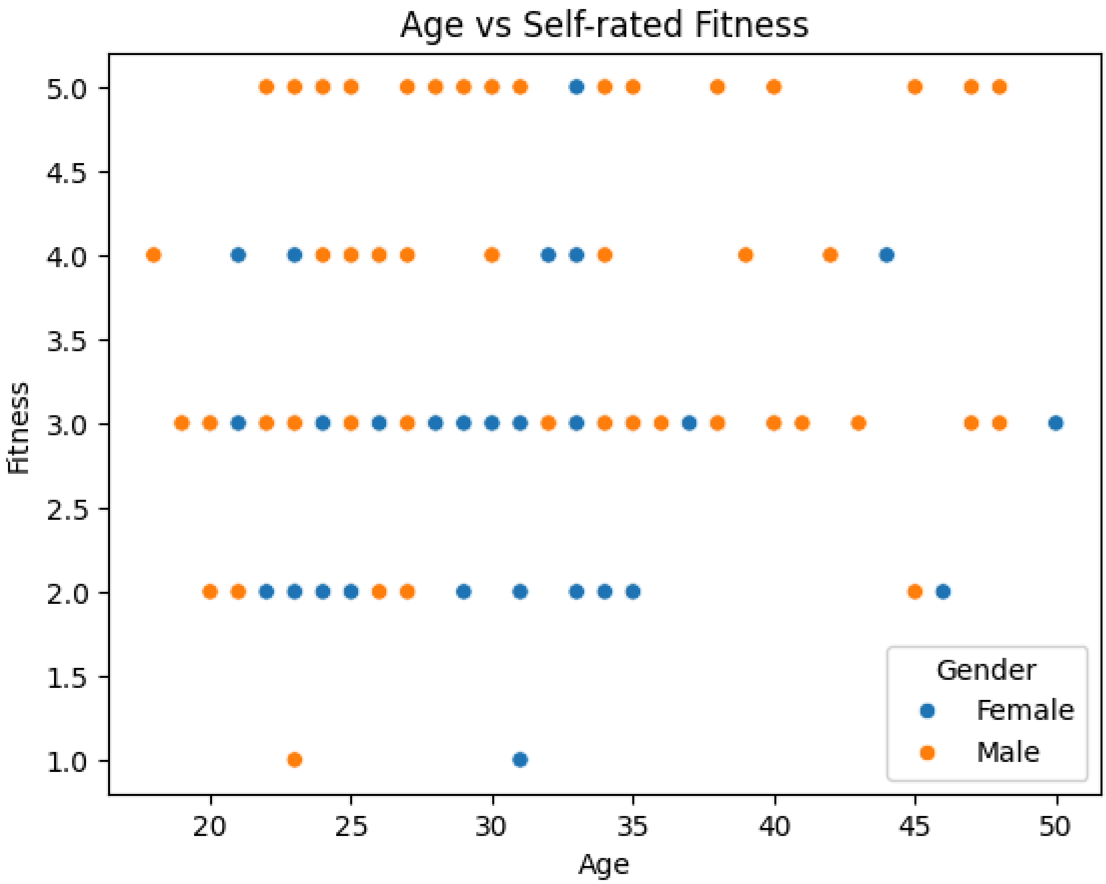


Observations :

- A significant 83.3% of the individuals in the dataset have a Bachelor's degree.
- The distribution is not balanced, with a heavy concentration in a single education category.
- Master’s degree holders make up 12.8%, which is the second most common group.
- Doctorate/Ph.D. holders are at 2.2%.
- High School education level is the least represented at 1.7%.

Visual Analysis - Bivariate

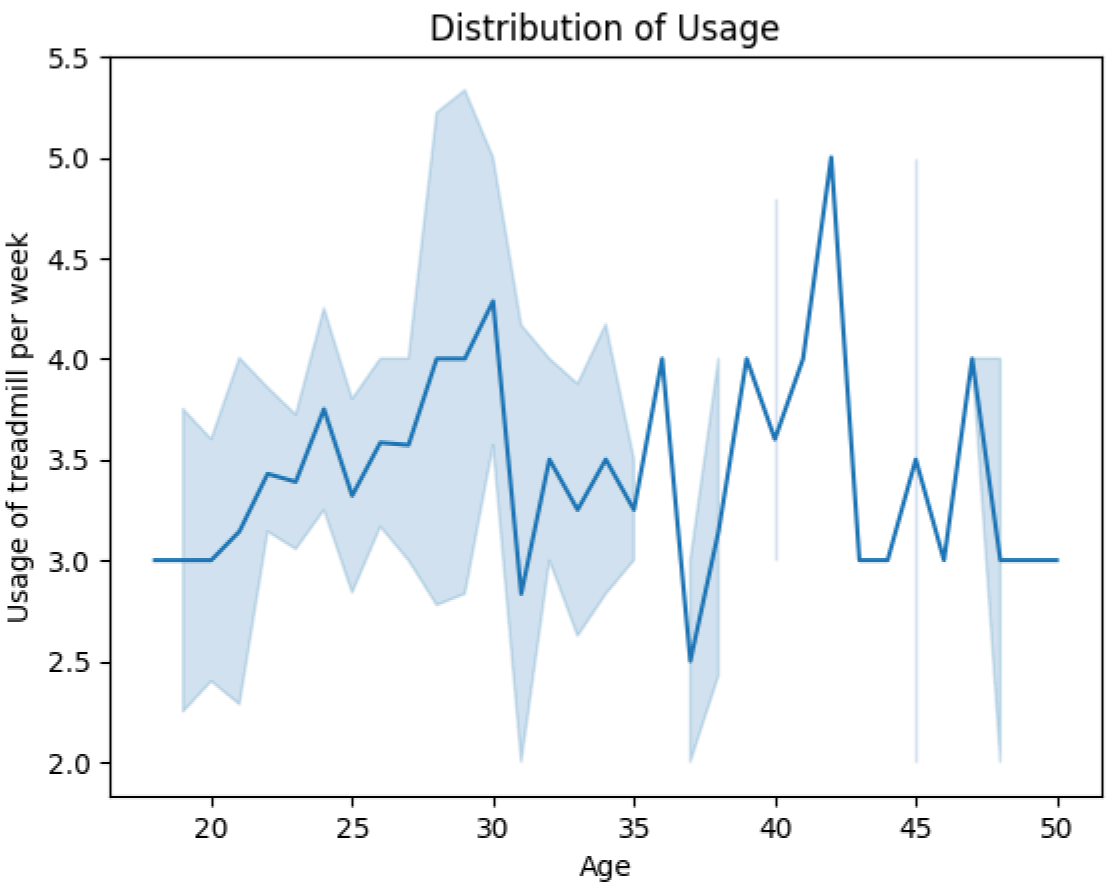
In [23]:
```
sns.scatterplot(data=data,x='Age',y='Fitness',hue='Gender')
plt.title('Age vs Self-rated Fitness')
plt.show()
```



Observations :

- The distribution appears fairly balanced, with most individuals rating their fitness level at 3.0 out of 5.0, suggesting a moderate level of fitness.
- Among the 180 individuals surveyed, the majority—approximately 45 females and 52 males—rated their fitness as 3, indicating an average level of weekly physical activity.
- The data also shows that most males rated their performance with the fitness product / self physical fitness as excellent (5 out of 5).

In [24]:
```
sns.lineplot(data=data,x='Age',y='Usage')
plt.xlabel('Age')
plt.ylabel('Usage of treadmill per week')
plt.title('Distribution of Usage')
plt.show()
```



Observations:

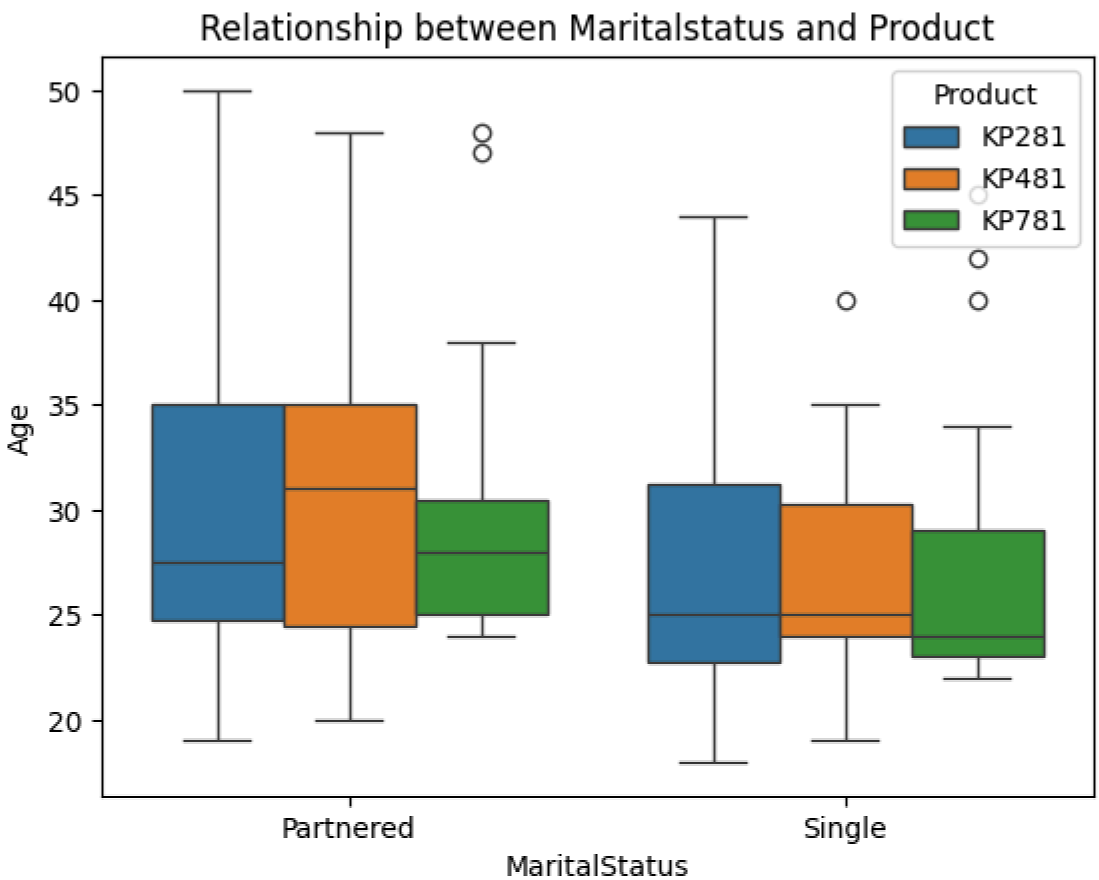
- The high surge in treadmill usage around age 30, where the average usage exceeds 4.5 times per week.
- There's a slight decline in treadmill usage after age 35.
- The shaded region (confidence interval) becomes wider after age 35–40, indicating greater variability in treadmill usage among older individuals.

In [25]:
```
sns.lineplot(data=data,x='Age',y='Miles',hue='Gender')
plt.title('Distribution of Miles of various Gender')
plt.show()
```

Observations :

- The distribution of miles ranges from 50 to 300 miles per week, with a consistent fluctuation observed between 50 and 200 miles, indicating varying activity levels within this range.
- Older individuals tend to be more physically active compared to their younger counterparts, as reflected in their higher treadmill usage.
- Males generally report higher mileage than females across most age groups, especially noticeable beyond age 30.

In [26]:
```
sns.boxplot(data=data,x='MaritalStatus',y='Age',hue='Product')
plt.title('Relationship between Maritalstatus and Product')
plt.show()
```

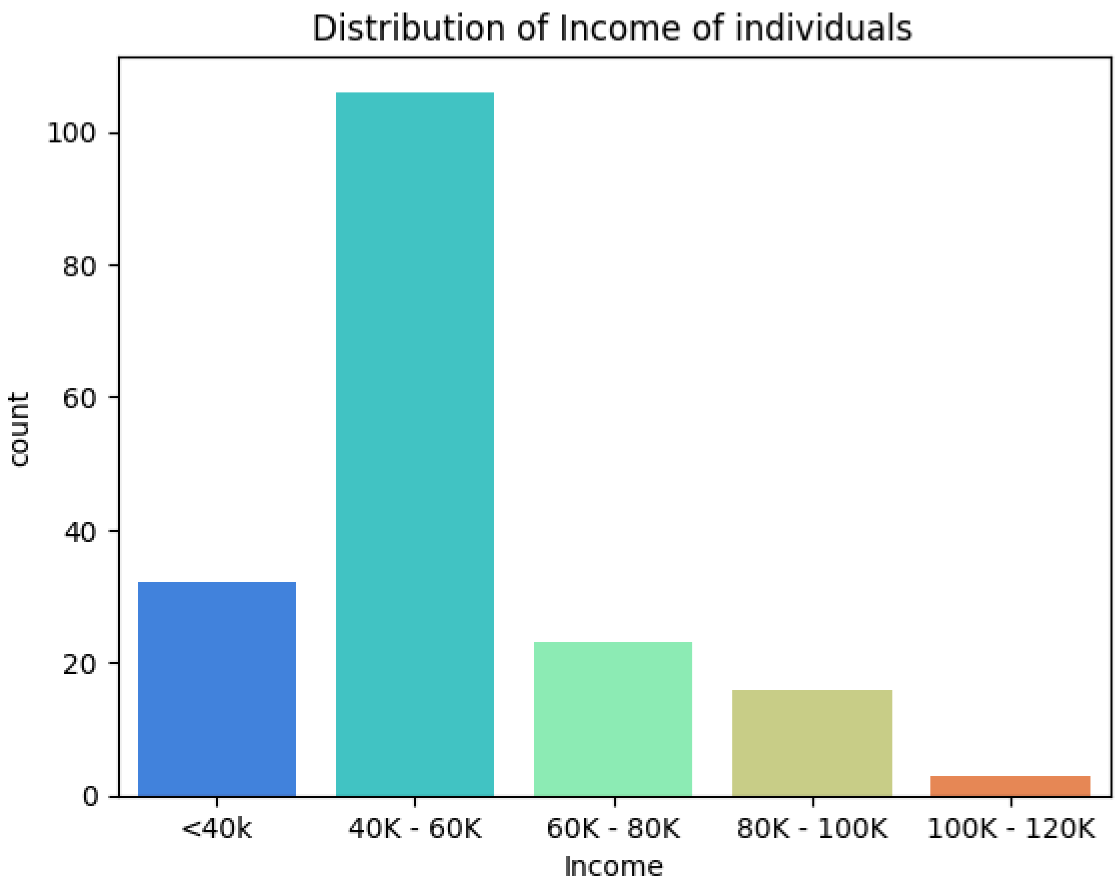


Observations :

- The product KP481 is slightly more popular among older individuals.
- KP281 and KP781 have a relatively broader age distribution.
- All three products show a narrower and more similar age distribution, mostly concentrated between ages 22 and 32.

In [27]:
```
bins = [20000 , 40000 , 60000 , 80000 , 100000 , 120000]
labels = ["<40k" , "40K - 60K" , "60K - 80K" , "80K - 100K" ,"100K - 120K" ]

Income_ranges = pd.cut(data["Income"] , bins = bins , labels = labels)
sns.countplot(x=Income_ranges, palette='rainbow')
plt.title("Distribution of Income of individuals")
plt.show()
```

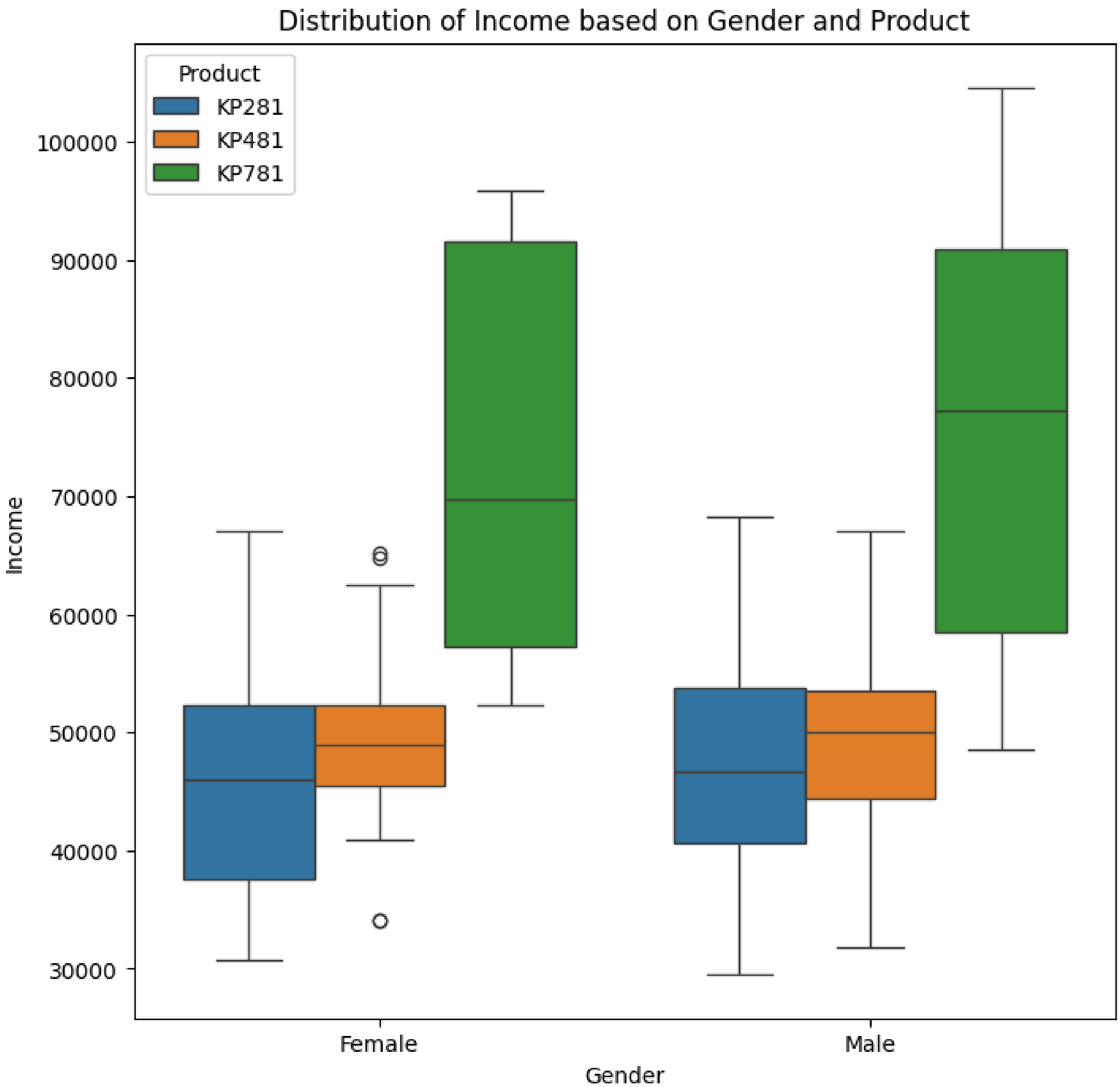
```
/tmp/ipython-input-27-3191038615.py:5: FutureWarning:

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 same e
ffect.

  sns.countplot(x=Income_ranges, palette='rainbow')
```



Observations:

- Most Individuals Earn 40k–60k: dominant income group, with the highest count (~100+)
- The 100k–120k range has very few individuals.

In [28]:
```
plt.figure(figsize=(8,8))
sns.boxplot(data=data,y="Income",x="Gender",hue="Product")
plt.title("Distribution of Income based on Gender and Product")
plt.show()
```



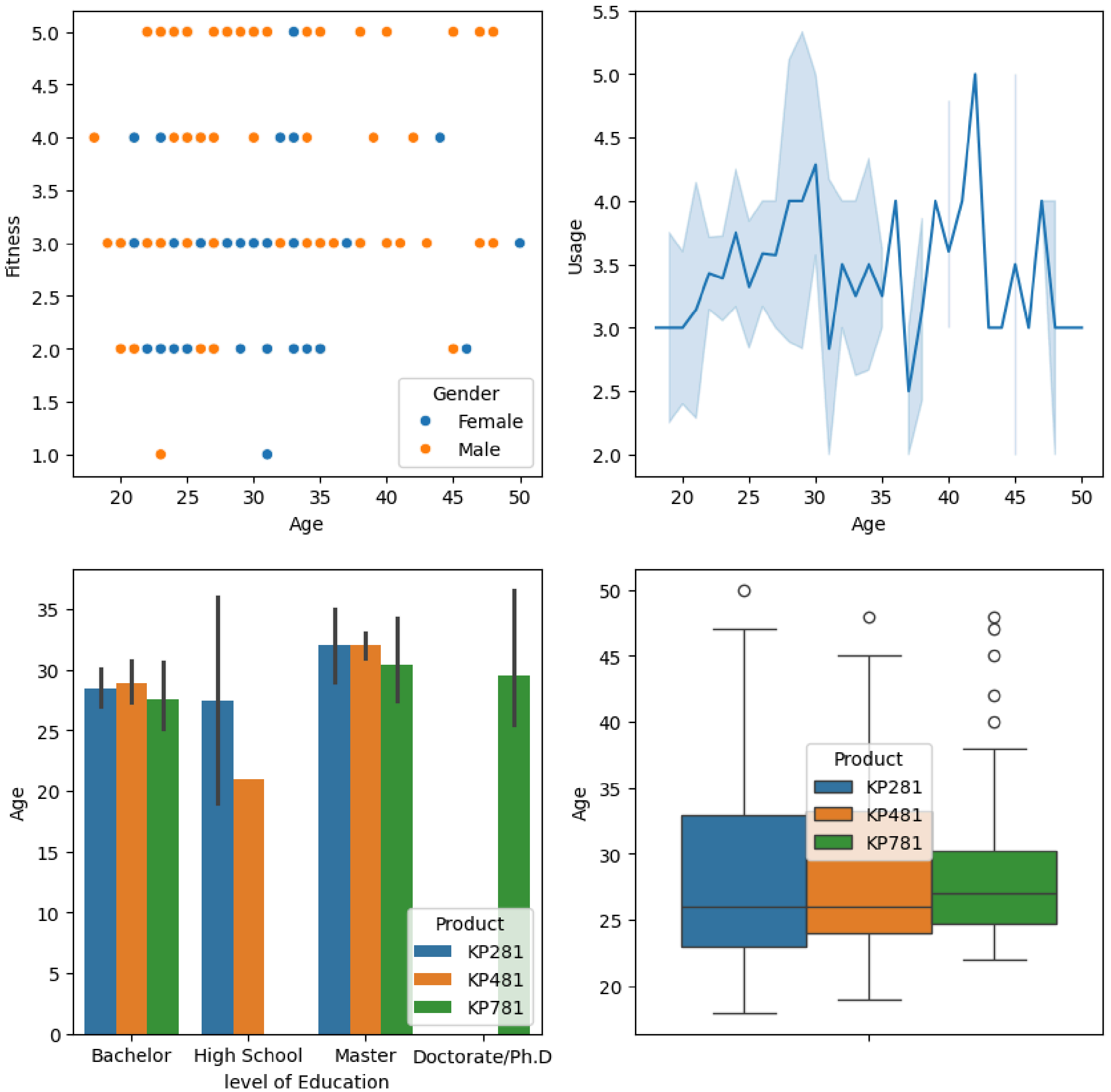
Observations:

- Highest Incomes are from Product KP781 - Both males and females using KP781 have the highest income range.
- For KP281 and KP481, the income distribution is fairly similar for both genders.
- KP781 has a large interquartile range (IQR), especially for males, suggesting more income variability.

Subplot

In [29]:
```
fig,ax=plt.subplots(2,2,figsize=(10,10))
```

```
sns.scatterplot(data=data,x='Age',y='Fitness',hue='Gender',ax=ax[0,0])
sns.lineplot(data=data,x='Age',y='Usage',ax=ax[0,1])
sns.barplot(data=data,x='level of Education',y='Age',hue='Product',ax=ax[1,0])
sns.boxplot(data=data,y='Age',hue='Product',ax=ax[1,1])
plt.show()
```



Observations :

1. **Fitness vs Age by Gender (Top-left)**

- Both genders show a wide spread of fitness scores across all ages.
- No strong trend with age, but most users report fitness levels between 3–5.

2. **Usage vs Age (Top-right)**

- Average product usage peaks around age 30, then stabilizes or slightly declines.
- There's higher variability in usage between ages 25–35, indicating mixed user behavior in this group.

3. **Education Level vs Age by Product (Bottom-left)**

- Master’s degree holders are slightly older than others (avg. age around 32).
- KP281 and KP481 products are used more consistently across all education levels.
- KP781 users with Doctorate degrees are younger (~29) compared to expectations.

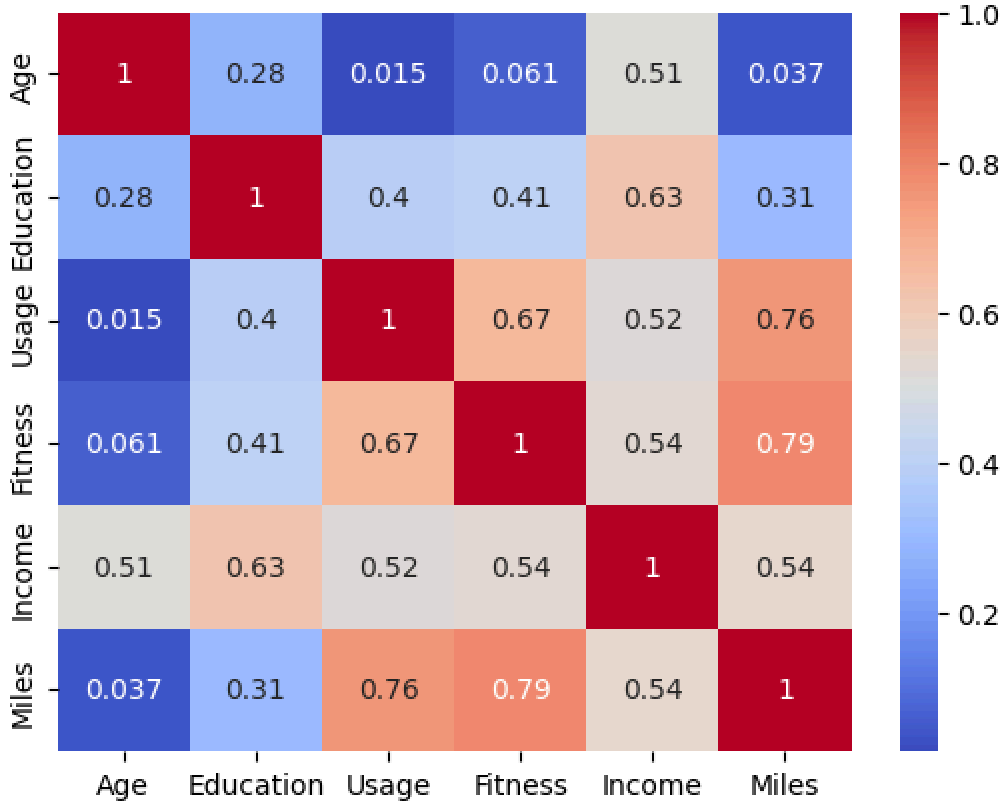
4. **Age Distribution by Product (Bottom-right)**

- KP281 has the widest age range (with several outliers above 45).
- KP481 users are mostly between 22–35 years.
- KP781 users tend to be younger, with a tighter age distribution.

correlation: Heatmaps, Pairplots

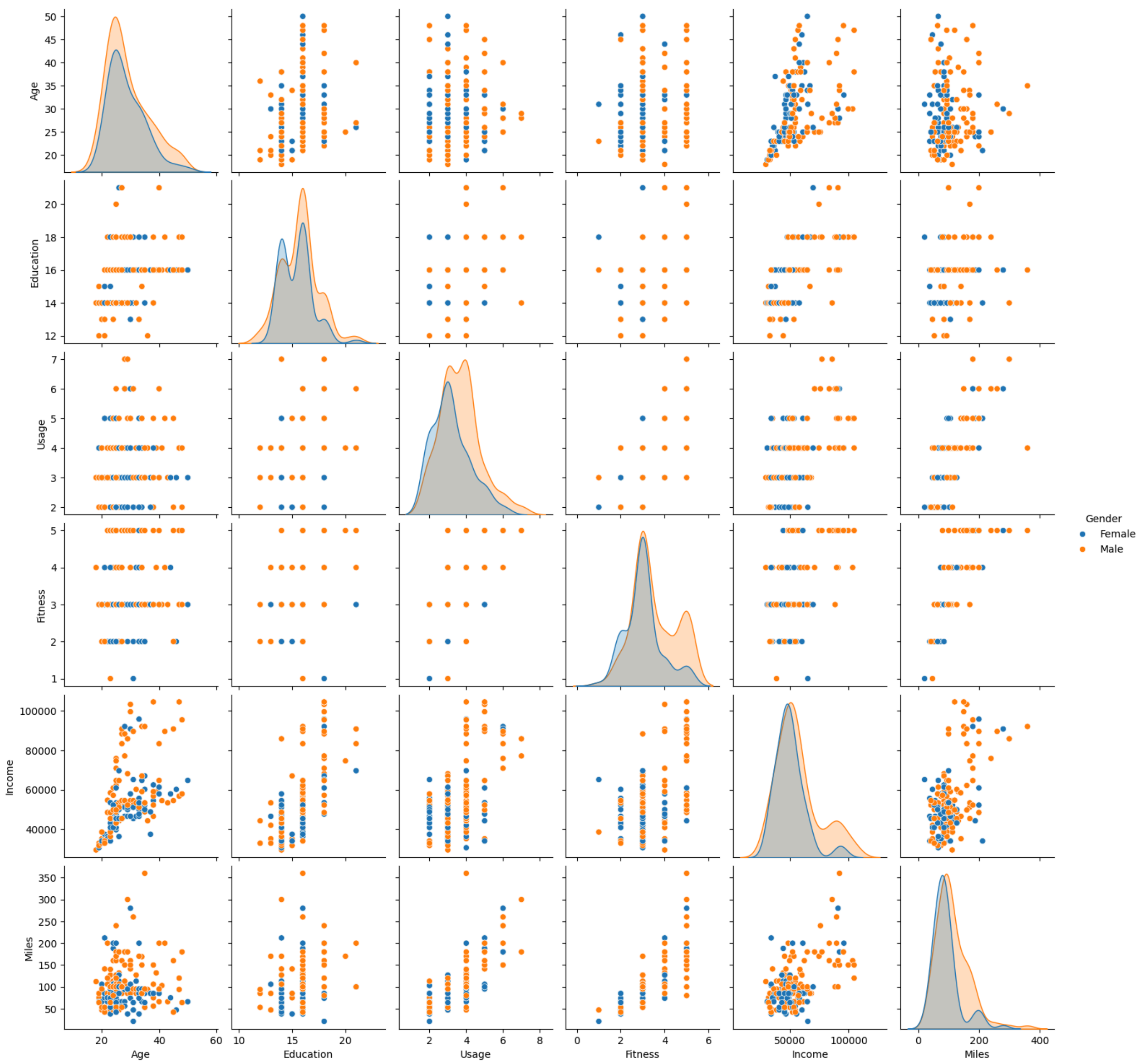
In [30]:
```
num_df = data.select_dtypes(include=['float64','int64']).corr()

sns.heatmap(num_df,annot=True,cmap='coolwarm')
```

Out[30]:
```
<Axes: >
```



In [31]:
```
sns.pairplot(data=data,hue='Gender')
```

Out[31]: <seaborn.axisgrid.PairGrid at 0x7c0255904e50>



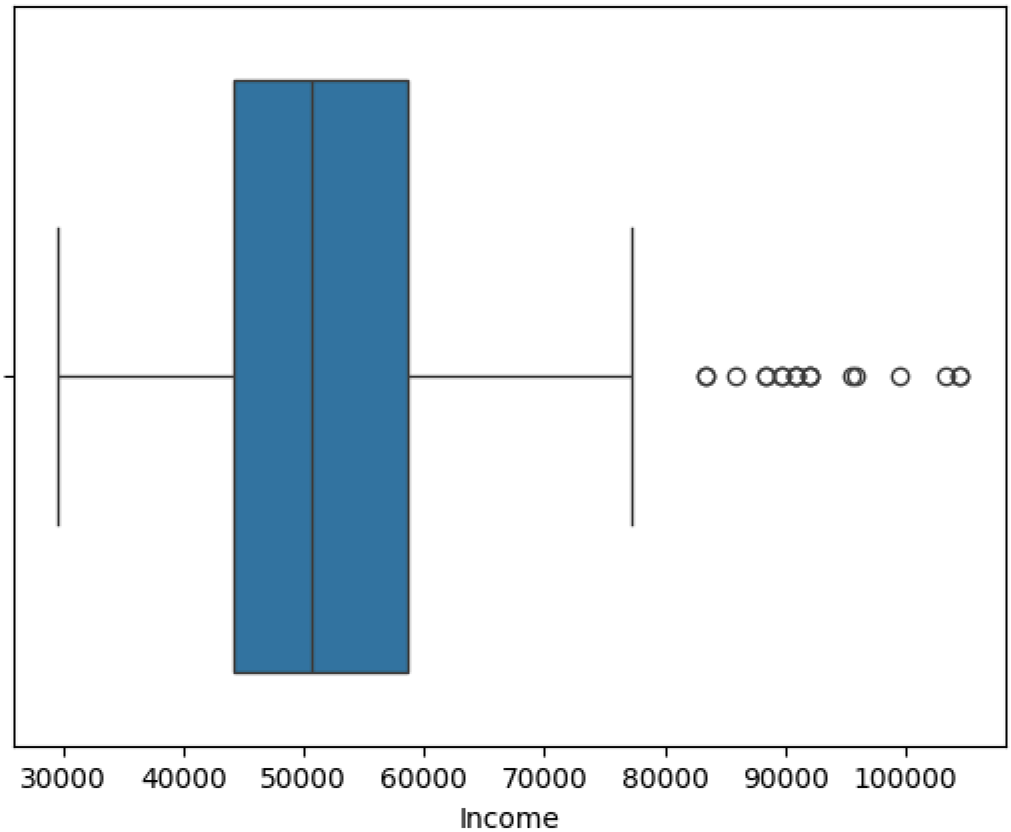
Detecting Outliers

In [32]:
```
df_income = data.copy()
df_income.head()
```

Out[32]:

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

In [33]:
```
sns.boxplot(data=df_income,x='Income')
plt.show()

Q1 = np.percentile(data['Income'],25)
Q3 = np.percentile(data['Income'],75)
IQR = Q3-Q1
print (f"The first quartile (25th percentile) for income is {Q1}")
print (f"The third quartile (75th percentile) for income is {Q3}")
print(f"The Interquartile range for income is {IQR}")

higher_bound_income=Q3+1.5*IQR
lower_bound_income=Q1-1.5*IQR
print(f"The higher bound for income is {higher_bound_income}")
print(f"The lower bound for income is {lower_bound_income}")
```




```
The first quartile (25th percentile) for income is 44058.75
The third quartile (75th percentile) for income is 58668.0
The Interquartile range for income is 14609.25
The higher bound for income is 80581.875
The lower bound for income is 22144.875
```

In [34]:
```
Outliers_income = df_income.loc[(df_income['Income']>higher_bound_income) | (c
print(f"The outliers present in income are {(Outliers_income)}")
```

```
The outliers present in income are [83416, 88396, 90886, 92131, 88396, 85906, 9
0886, 103336, 99601, 89641, 95866, 92131, 92131, 104581, 83416, 89641, 90886, 1
04581, 95508]
```

In [35]:
```
##Removing the Outliers - this maintains only the values within the range - re
```

```
df_cleaned_income=df_income[(df_income["Income"]>=lower_bound_income)&(df_inco
df_cleaned_income
```

Out[35]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income
0	KP281	18	Male	14	Single	3	4	29562
1	KP281	19	Male	15	Single	2	3	31836
2	KP281	19	Female	14	Partnered	4	3	30699
3	KP281	19	Male	12	Single	3	3	32973
4	KP281	20	Male	13	Partnered	4	2	35247
...	...	...	...	...	...	...	...	...
156	KP781	25	Male	20	Partnered	4	5	74701
157	KP781	26	Female	21	Single	4	3	69721
158	KP781	26	Male	16	Partnered	5	4	64741
163	KP781	28	Male	18	Partnered	7	5	77191
165	KP781	29	Male	18	Single	5	5	52290

161 rows × 10 columns

In [36]:

```
Q5 = np.percentile(data['Income'],5)
Q95 = np.percentile(data['Income'],95)
print (f"The first quartile (5th percentile) for income is {Q5}")
print (f"The third quartile (95th percentile) for income is {Q95}")
```

```
The first quartile (5th percentile) for income is 34053.15
The third quartile (95th percentile) for income is 90948.24999999999
```

In [37]:

```
#Reduces the impact of extreme outliers on models and stats - by using np.clip
df_income["Income"]=np.clip(df_income["Income"], Q5,Q95)
df_income
```

Out[37]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income
0	KP281	18	Male	14	Single	3	4	34053.15
1	KP281	19	Male	15	Single	2	3	34053.15
2	KP281	19	Female	14	Partnered	4	3	34053.15
3	KP281	19	Male	12	Single	3	3	34053.15
4	KP281	20	Male	13	Partnered	4	2	35247.00
...	...	...	...	...	...	...	...	...
175	KP781	40	Male	21	Single	6	5	83416.00
176	KP781	42	Male	18	Single	5	4	89641.00
177	KP781	45	Male	16	Single	5	5	90886.00
178	KP781	47	Male	18	Partnered	4	5	90948.25
179	KP781	48	Male	18	Partnered	4	5	90948.25

180 rows × 10 columns

In [38]:

```
print(f"The total Outliers clipped :{data.shape[0] - df_cleaned_income.shape[0
```

```
The total Outliers clipped :19
```

Why Clipping is Better in Your Case:

- You only have 180 rows → removing 19 rows (~10.5%) can weaken your model or summary statistics
- The outliers (like 83k, 90k, 104k income) look real, not data errors

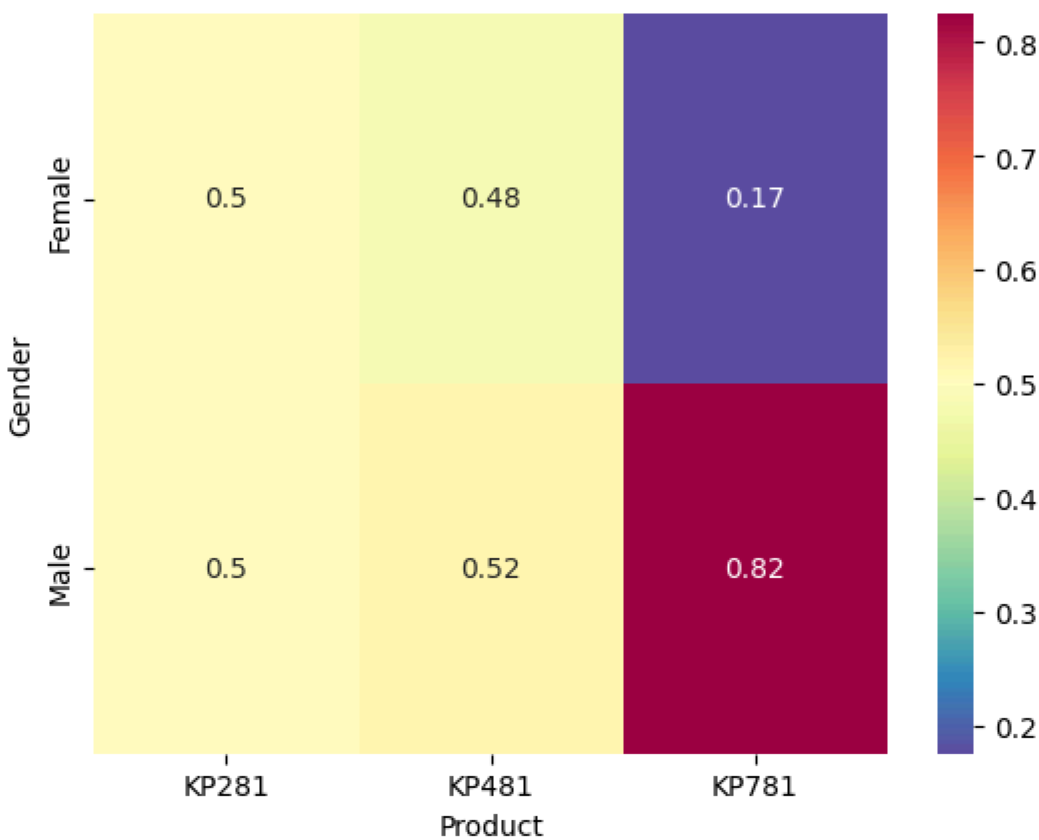
Conditional Probability

What is the probability of a male customer buying a KP781 treadmill ?

In [39]:
```python
male_data = data[(data["Product"]=="KP781") & (data["Gender"]=="Male")]
length_male = len(male_data)
print("favorable outcomes :" , length_male)

product_data = data[data["Product"]=="KP781"]
length_product = len(product_data)
print("Possible outcomes of KP781 :" ,length_product)

probability = length_male/length_product
print("Probability of a male customer buying a KP781 treadmill:",probability)
```

```
favorable outcomes : 33
Possible outcomes of KP781 : 40
Probability of a male customer buying a KP781 treadmill: 0.825
```

In [40]:
```python
sns.heatmap(pd.crosstab(data['Gender'],data['Product'],normalize='columns'),ar
plt.show()
```



Conditional Probability P(Gender|Product)

The Probability of Male customer given that he is buying KP281,P(Customer=Male|Product = KP281)=0.5

The Probability of Male customer given that he is buying KP481,P(Customer=Male|Product = KP481)=0.52

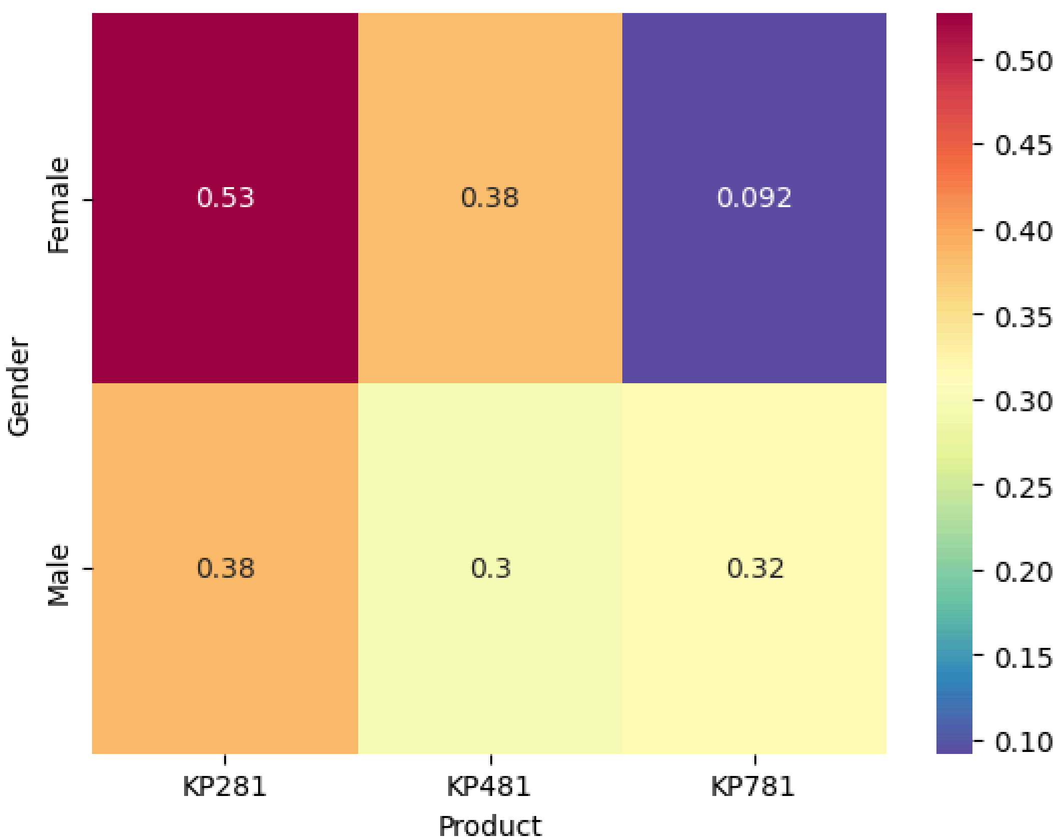
The Probability of Male customer given that he is buying KP781,P(Customer=Male|Product = KP781)=0.82

The Probability of Female customer given that she is buying KP281,P(Customer=Female|Product = KP281)=0.5

The Probability of Female customer given that she is buying KP481,P(Customer=Female|Product = KP481)=0.48

The Probability of Female customer given that she is buying KP781,P(Customer=Female|Product = KP781)=0.17

In [41]:
```
sns.heatmap(pd.crosstab(data['Gender'],data['Product'],normalize='index'),anno
plt.show()
```



Conditional Probability P(Product|Gender)

The Probability of buying KP281 and given that the customer is Male,P(Product = KP281|Customer=Male)=0.38

The Probability of buying KP481 and given that the customer is Male,P(Product = KP481|Customer=Male)=0.3

The Probability of buying KP781 and given that the customer is Male,P(Product =

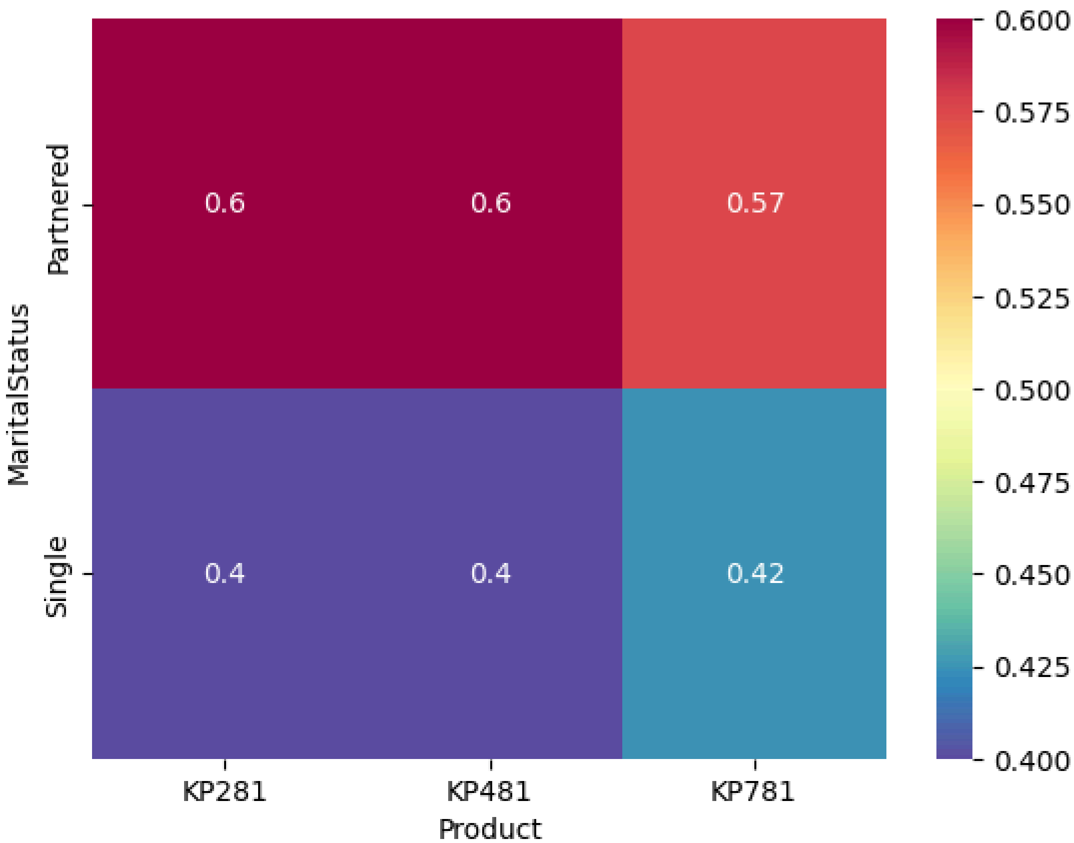
KP781|Customer=Male)=0.32

The Probability of buying KP281 and given that the customer is Female,P(Product = KP281|Customer=Female)=0.53

The Probability of buying KP481 and given that the customer is Female,P(Product = KP481|Customer=Female)=0.38

The Probability of buying KP781 and given that the customer is Female,P(Product = KP781|Customer=Female)=0.092

In [42]:
```
sns.heatmap(pd.crosstab(data['MaritalStatus'],data['Product'],normalize='colum
plt.show()
```



Conditional Probability P(Maritalstatus|Product)

The Probability of customer is Single and given that he/she is buying KP281,P(Maritalstatus=Single|Product = KP281)=0.4

The Probability of customer is Single and given that he/she is buying KP481,P(Maritalstatus=Single|Product = KP481)=0.4

The Probability of customer is Single and given that he/she is buying

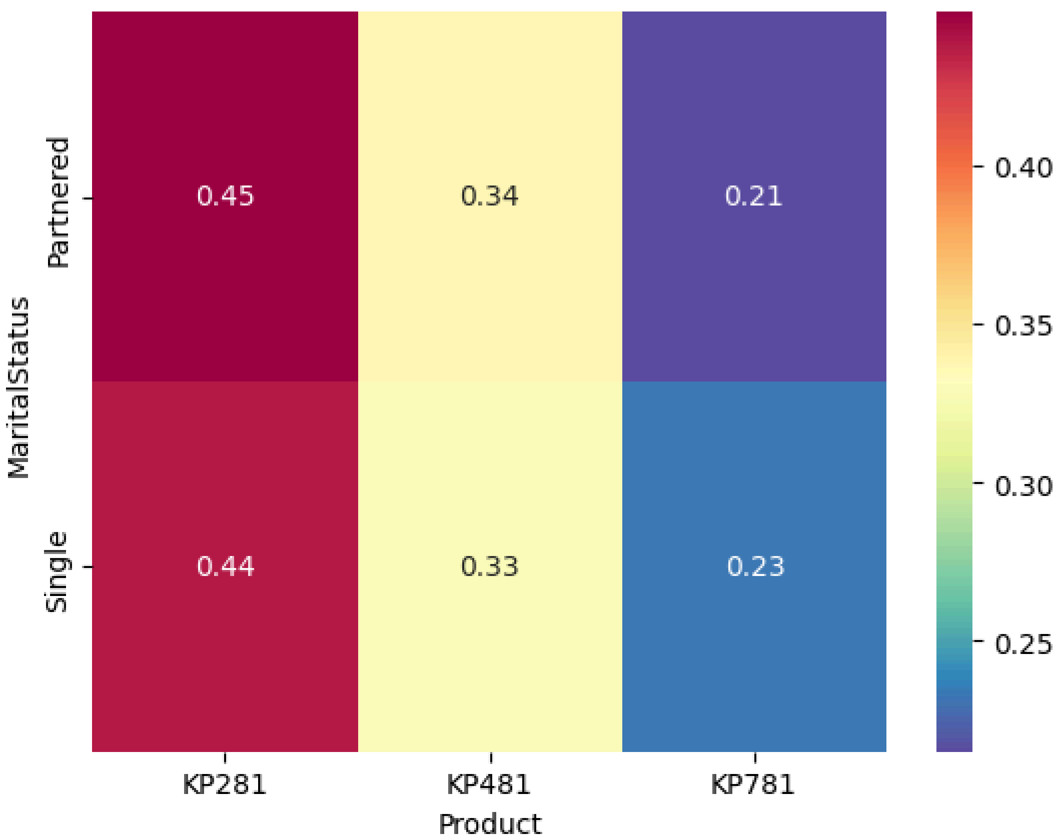
KP781,P(Maritalstatus=Single|Product = KP781)=0.42

The Probability of customer is Partnered and given that he/she is buying KP281,P(Maritalstatus=Partnered|Product = KP281)=0.6

The Probability of customer is Partnered and given that he/she is buying KP481,P(Maritalstatus=Partnered|Product = KP481)=0.6

The Probability of customer is Partnered and given that he/she is buying KP781,P(Maritalstatus=Partnered|Product = KP781)=0.57

In [43]:
```
sns.heatmap(pd.crosstab(data['MaritalStatus'],data['Product'],normalize='index
plt.show()
```



Conditional Probability P(Product|Maritalstatus)

The Probability of buying KP281 and given that the customer is Single, P(Product=KP281|Maritalstatus=Single) = 0.44

The Probability of buying KP481 and given that the customer is Single, P(Product=KP481|Maritalstatus=Single) = 0.33

The Probability of buying KP781 and given that the customer is Single, P(Product=KP781|Maritalstatus=Single) = 0.23

The Probability of buying KP281 and given that the customer is Partnered, P(Product=KP281|Maritalstatus=Partnered) = 0.45

The Probability of buying KP481 and given that the customer is Partnered, P(Product=KP481|Maritalstatus=Partnered) = 0.34

The Probability of buying KP781 and given that the customer is Partnered, P(Product=KP781|Maritalstatus=Partnered) = 0.21

Marginal Probabilities

In [44]:
```
data["Product"].value_counts(normalize=True)
```

Out[44]:

Product	proportion
KP281	0.444444
KP481	0.333333
KP781	0.222222

dtype: float64

Observations:

- KP281 is the best selling model , it may have the best combination of price, features, and market fit.
- KP781 is least purchased due to high price, advanced features not needed.

In [45]:
```
data["Gender"].value_counts(normalize=True)
```

Out[45]:

Gender	proportion
Male	0.577778
Female	0.422222

dtype: float64

Observations:

- AeroFit treadmills are more popular with male buyers overall.
- Female customers are still a significant segment (≈ 42%)

In [46]:
```python
data["MaritalStatus"].value_counts(normalize=True)
```

Out[46]:

MaritalStatus	proportion
Partnered	0.594444
Single	0.405556

dtype: float64

In [47]:
```python
data["Fitness"].value_counts(normalize=True)
```

Out[47]:

Fitness	proportion
3	0.538889
5	0.172222
2	0.144444
4	0.133333
1	0.011111

dtype: float64

Customer Profiling

�♂ KP281 — Budget-Conscious, Young Adults

Customer Profile:

Age Group: 18–50 years

Gender: Balanced share between male and female customers

Income Level: Low to mid-income (₹30,000–₹65,000 per month)

Primary Motivation: Entry-level fitness and staying active at home

Lifestyle: Students, early-career professionals, and young couples

Usage Pattern: Typically used 3 times per week on average

Key Traits: Value-driven, prefers ease of use and space-saving features

Marketing Strategy: Highlight affordability.

��♂ KP481 — Performance-Oriented Mid-Life Professionals

Customer Profile:

Age Group: 19–50 years

Gender: Balanced share between male and female customers

Income Level: Low to mid-income (₹45,000–₹68,000 per month)

Primary Motivation: Entry-level fitness and staying active at home

Lifestyle: Bachelor's Students,Marriage couples,Working professionals.

Usage Pattern: Typically used 3 times per week on average

Key Traits: Value-driven, prefers ease of use and space-saving features

---

��♂ KP781 — Premium, High-Income, Senior Wellness Seekers

Customer Profile:

Age Group: 30 - 50 + years

Gender: Predominantly male

Income Level: Low to mid-income (₹70,000–₹1,00,000 per month)

Primary Motivation: Health Management

Lifestyle: Master's Students,Marriage couples, Experienced Professionals.

Usage Pattern: Typically used 4 times per week on average

Key Traits: Interested in comfort and brand prestige

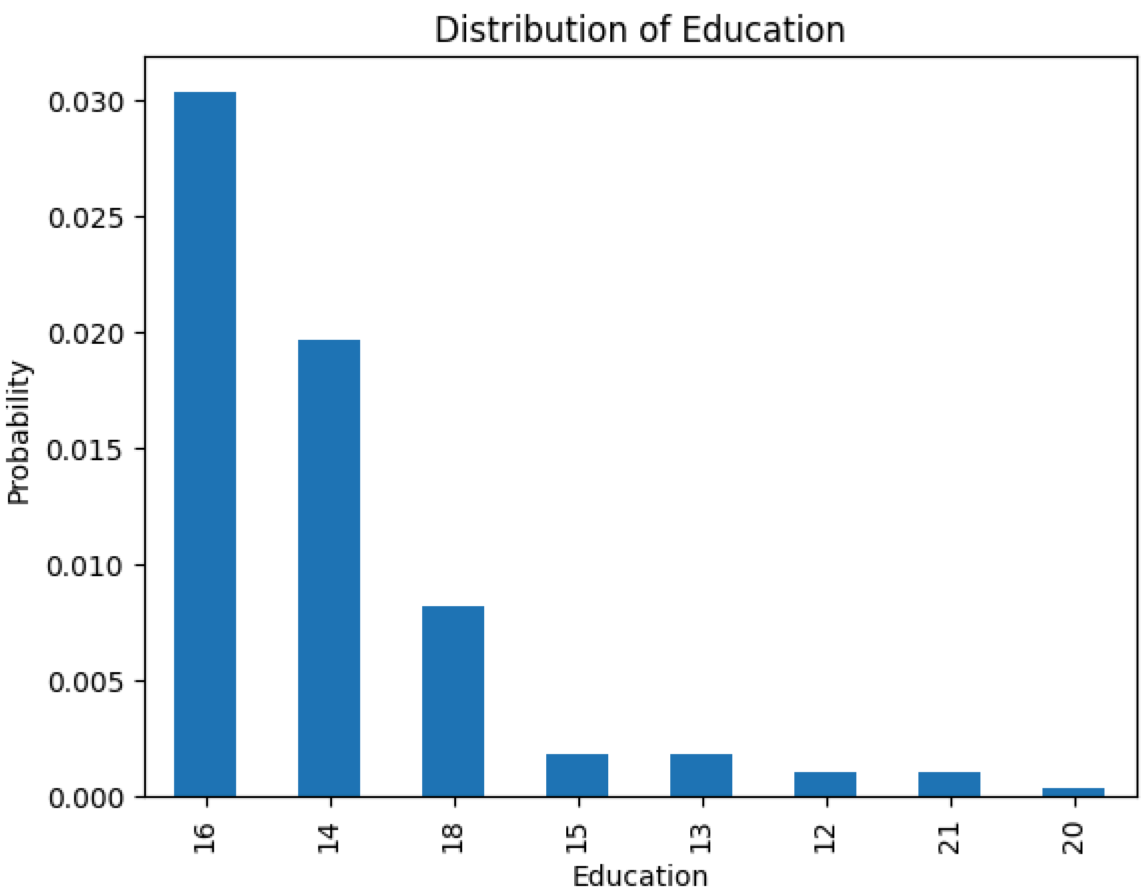
Distributions:

Probability Mass Function

- PMF ---> discrete variable

- Let's plot the distribution of individuals based on their years of education.

In [48]:
```
Edu_data = data["Education"].sum()
Edu_data

single_data = data["Education"].value_counts()
single_data

prob = single_data/Edu_data
prob.plot(kind='bar')
plt.xlabel("Education")
plt.ylabel("Probability")
plt.title("Distribution of Education")
plt.show()
```



Observations:

- Education Level 16 Dominates: The highest probability is for education level 16, meaning it is the most common in the dataset.
- Other Common Levels: Education levels 14 and 18 also appear

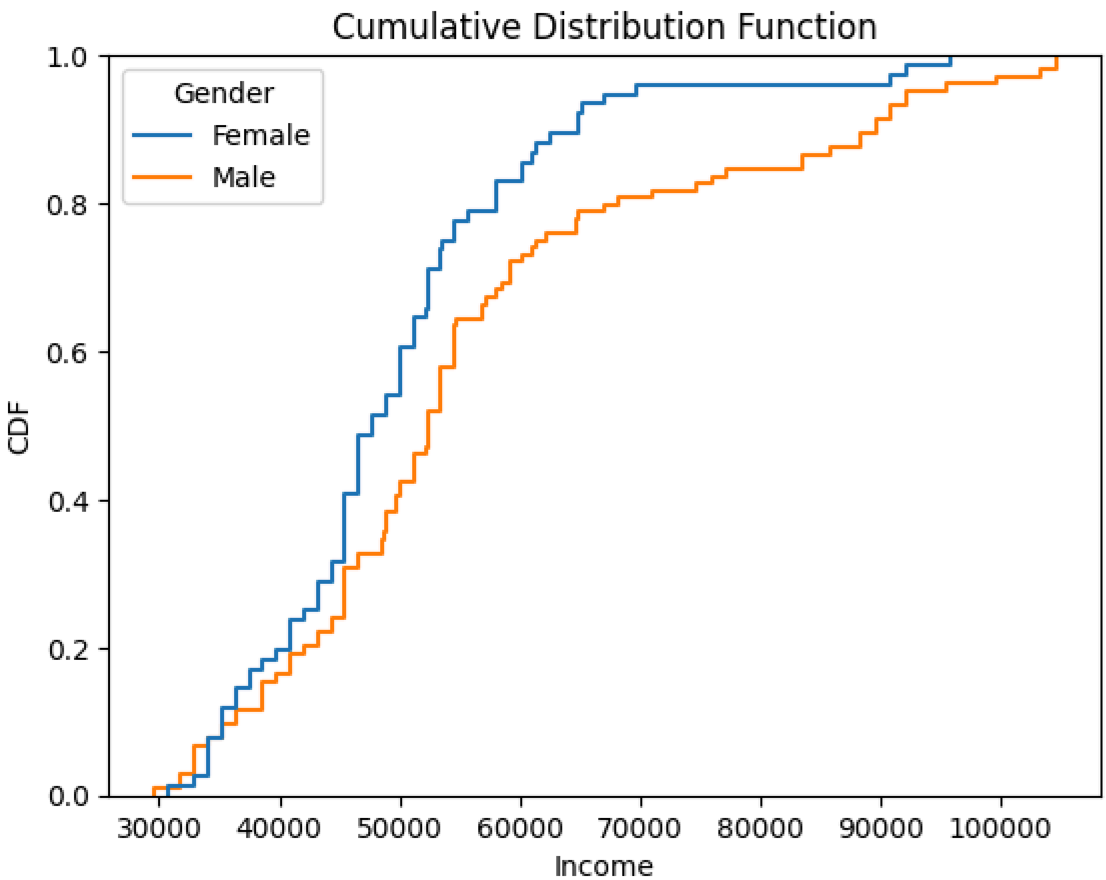
frequently, suggesting they are relatively prevalent.

- Rare Education Levels: Levels 15, 13, 12, 21, and 20 have significantly lower probabilities, indicating they are less common.

Cumulative Distribution Function

CDF - For continuous data

In [49]:
```
sns.ecdfplot(data=data, hue='Gender', x='Income')
plt.xlabel("Income")
plt.ylabel("CDF")
plt.title("Cumulative Distribution Function")
plt.show()
```

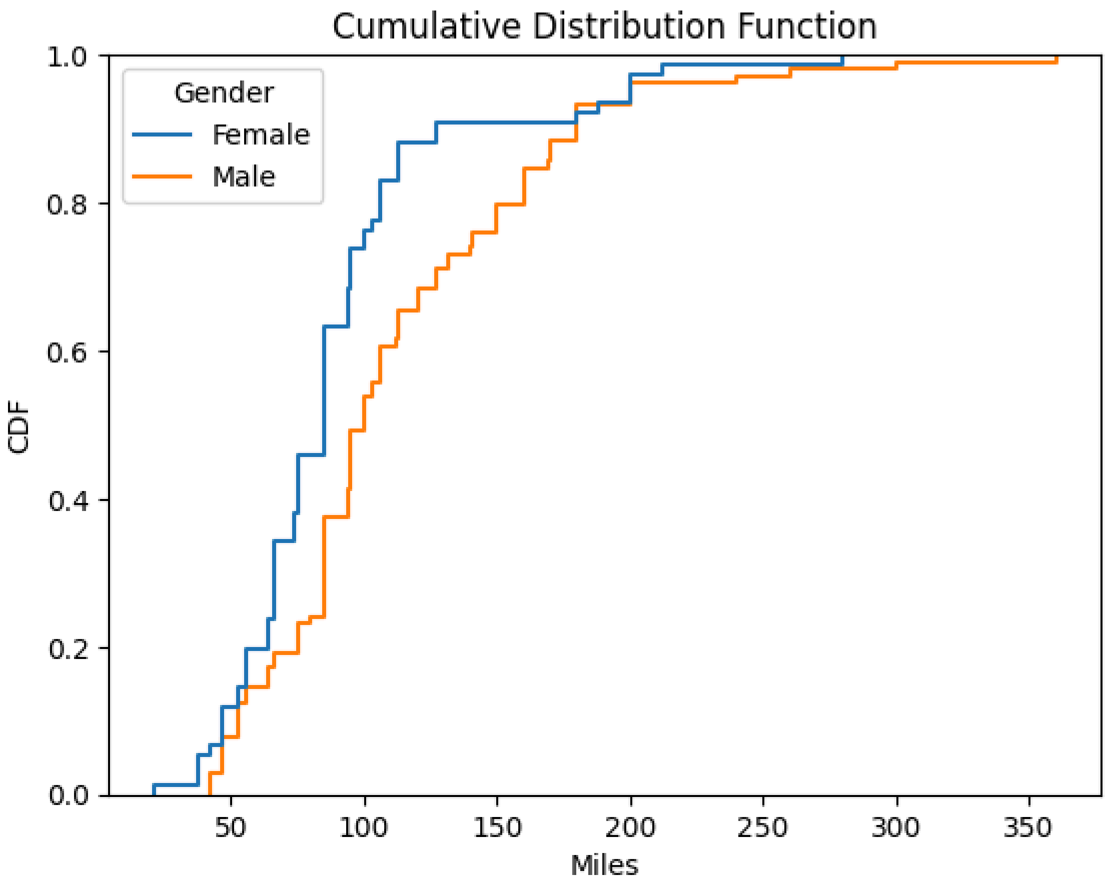


Observations:

- Males appear to have a more gradual increase in their cumulative distribution, indicating that they are more likely to be earning higher salaries than females.

- A significant portion of female income seems to be concentrated in the lower ranges, whereas males have a wider distribution across higher incomes.

In [50]:
```
sns.ecdfplot(data=data,x="Miles",hue="Gender")
plt.xlabel("Miles")
plt.ylabel("CDF")
plt.title("Cumulative Distribution Function")
plt.show()
```



Observations:

- A significant portion of female mile seems to be concentrated in the lower ranges, whereas males have a wider distribution across higher miles.

In [53]:
```
data.head()
```

Out[53]:

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

In [58]:
```
data["level of Education"].unique()
```

Out[58]:
```
array(['Bachelor', 'High School', 'Master', 'Doctorate/Ph.D'],
      dtype=object)
```

Converting categorical to numerical & Correlating the product with all the fields

In [51]:
```
df_copy=data.copy()
```

In [60]:
```
df_copy['Gender'].replace(['Male', 'Female'], [1,0], inplace= True)
df_copy['Product'].replace(['KP281', 'KP481', 'KP781'], [0,1,2], inplace= True
df_copy['MaritalStatus'].replace(['Single', 'Partnered'], [0,1], inplace= True
df_copy['level of Education'].replace(['High School','Bachelor','Master','Doct
```

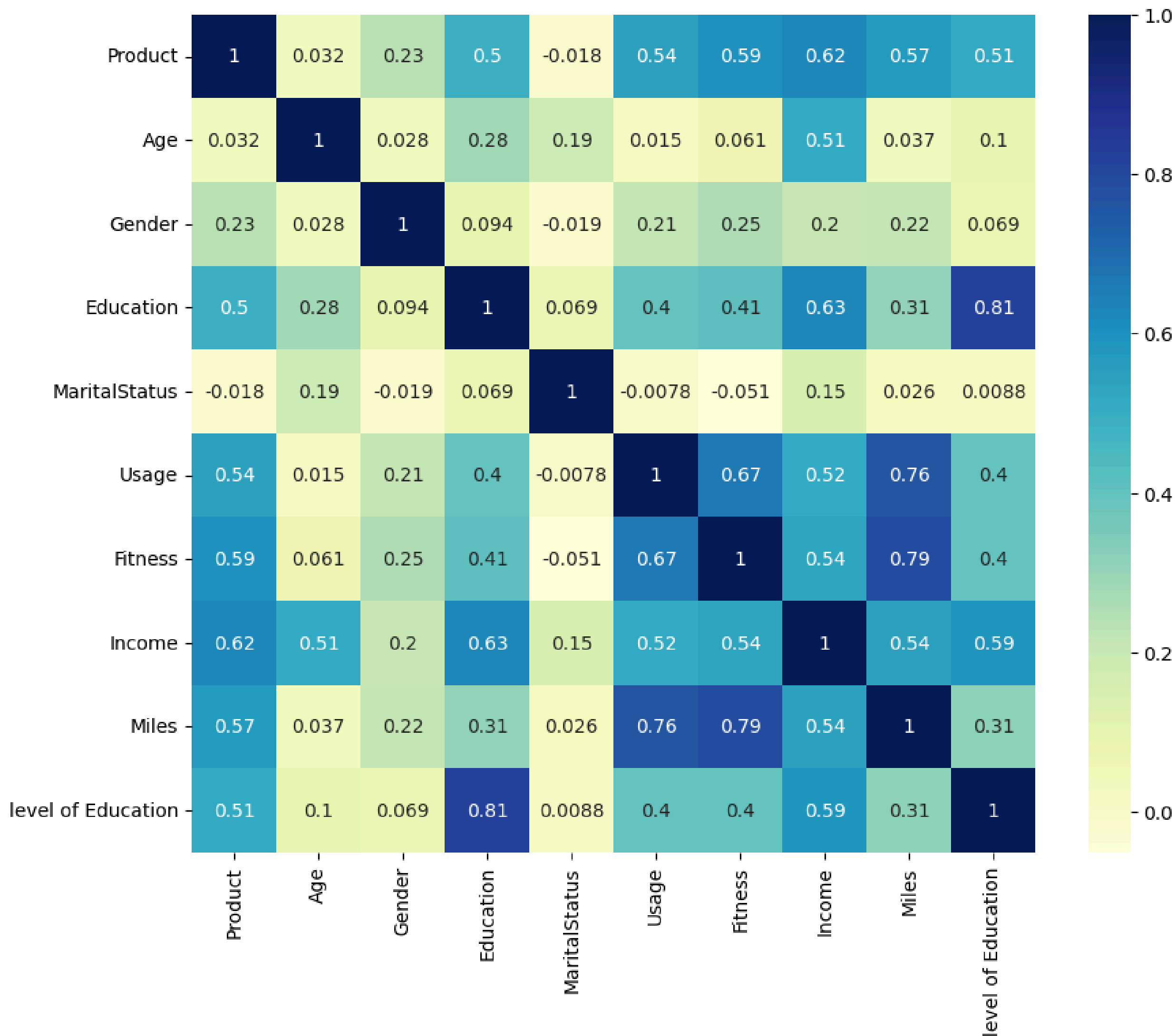
```
/tmp/ipython-input-60-90036377.py:1: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work bec
ause the intermediate object on which we are setting values always behaves as a
copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.me
thod({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, t
o perform the operation inplace on the original object.


  df_copy['Gender'].replace(['Male', 'Female'], [1,0], inplace= True)
/tmp/ipython-input-60-90036377.py:2: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work bec
ause the intermediate object on which we are setting values always behaves as a
copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.me
thod({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, t
o perform the operation inplace on the original object.


  df_copy['Product'].replace(['KP281', 'KP481', 'KP781'], [0,1,2], inplace= Tru
e)
/tmp/ipython-input-60-90036377.py:3: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work bec
ause the intermediate object on which we are setting values always behaves as a
copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.me
thod({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, t
o perform the operation inplace on the original object.


  df_copy['MaritalStatus'].replace(['Single', 'Partnered'], [0,1], inplace= Tru
e)
/tmp/ipython-input-60-90036377.py:4: FutureWarning: A value is trying to be set
on a copy of a DataFrame or Series through chained assignment using an inplace
method.
The behavior will change in pandas 3.0. This inplace method will never work bec
ause the intermediate object on which we are setting values always behaves as a
copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.me
thod({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, t
o perform the operation inplace on the original object.


  df_copy['level of Education'].replace(['High School','Bachelor','Master','Doc
```

```
torate/Ph.D'],[1,2,3,4],inplace=True)
/tmp/ipython-input-60-90036377.py:4: FutureWarning: Downcasting behavior in `re
place` is deprecated and will be removed in a future version. To retain the old
behavior, explicitly call `result.infer_objects(copy=False)`. To opt-in to the
future behavior, set `pd.set_option('future.no_silent_downcasting', True)`
  df_copy['level of Education'].replace(['High School','Bachelor','Master','Doc
torate/Ph.D'],[1,2,3,4],inplace=True)
```

In [66]:

```
plt.figure(figsize=(10,8))
sns.heatmap(df_copy.corr(), cmap="YlGnBu", annot = True)
```

Out[66]: <Axes: >



Noteworthy Points

- The product/treadmill purchased highly correlates with Education, Income, Usage, Fitness and Miles.

- Age is highly correlated to Income (0.51) which definitely seems reasonable. It’s also correlated with Education and Marital Status which stands completely alright.
- Gender certainly has some correlation to Usage, Fitness, Income and Miles.
- Education is correlated to Age and Miles. It’s highly correlated to Income
(as expected). It’s sufficiently correlated to Usage and Fitness too.
- Marital Status has some correlation to Income and Age (as expected).
- Usage is extremely correlated to Fitness and Miles and has a higher correlation with Income as well.
- Fitness has a great correlation with Income.

More Observations and Possibilities:

- Product, Fitness, Usage and Miles depict a ridiculously higher correlation among themselves which looks as expected since more the usage implies more miles run and certainly more fitness.
- Also a story which seems reasonable is that Age and Education (predominately) are indicators of Income which affects the products bought. The more advanced the product is, the more its usage and hence more the miles run which in turn improves the fitness.

Recommendations :

- **Product Analysis**:
- Focus marketing efforts and product development on KP281 and KP481, especially targeting users with annual incomes between 30k–70k USD.
- Implement targeted discounts and promotional campaigns for the KP781 product to expand its reach beyond high-income users.

- **Gender Difference :**
- Launch wellness campaigns, organize fitness challenges, and provide

educational content aimed at encouraging more active lifestyles among women.

- Develop inclusive marketing materials, offer personalized features, and build partnerships with female influencers or health professionals.

- **level of Education:**
- Partner with schools to provide fitness workshops, student-friendly treadmill models, and subsidized packages.
- Launch tailored marketing campaigns through college partnerships, fitness challenges, and student discounts to strengthen brand presence in this demographic.

Maritalstatus Analysis:

- Launch couple- or group-based fitness challenges, referral programs, or family-friendly workout packages to build on this trend.
- Sales data indicates that KP481 and KP781 have lower adoption rates among single individuals - Offer solo fitness plans, personalized goal-setting tools

In [50]:

```

```