1.1 Problem Statement

Aerofit is a leading manufacturer of fitness equipment in India. They boast of a wide range of machines in their lineup including treadmills, elliptical trainers etc. across residential and commercial customers. They have both online and offline presence.

The goal of this analysis is to create a customer profile for each of the product in the treadmill category. This will help Aerofit in understanding where to concentrate their marketing efforts and to better anticipate the needs of new customers.

In [1]:

```
#Importing all the necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
#Reading the dataset and creating a Pandas DataFrame
df=pd.read_csv('https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/125/original/aerofit_treadmill.csv?1639992749')
```

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

1.2 Primary Analysis

In [4]:

```
df.shape
```

Out[4]:

```
(180, 9)
```

In [5]:

```
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 [6]:

```
df.describe()
```

Out[6]:

	Age	Education	Usage	Fitness	Income	Miles
count	180.000000	180.000000	180.000000	180.000000	180.000000	180.000000
mean	28.788889	15.572222	3.455556	3.311111	53719.577778	103.194444
std	6.943498	1.617055	1.084797	0.958869	16506.684226	51.863605
min	18.000000	12.000000	2.000000	1.000000	29562.000000	21.000000
25%	24.000000	14.000000	3.000000	3.000000	44058.750000	66.000000
50%	26.000000	16.000000	3.000000	3.000000	50596.500000	94.000000
75%	33.000000	16.000000	4.000000	4.000000	58668.000000	114.750000
max	50.000000	21.000000	7.000000	5.000000	104581.000000	360.000000

1. The dataset has 180 rows and 9 columns

2. 6 columns are of integer data type and 3 of object data type

3. There are no missing values in any of the columns

4. The statistical analysis gives the following insights about the customers

4.1 Their age ranges from 18 to 50 with a mean age of 28.78

4.2 The clientele is fairly young with 75% within the age of 33

4.3 They have an average of 15.57 years of education

4.4 They understand that there needs more improvement in their fitness levels with a mean fitness level of

2. Non Graphical Analysis

Finding marginal probabilities of each unique value for the relevant columns

In [71]:

```
round((df["Product"].value_counts())/len(df),2)
```

Out[71]:

```
KP281    0.44
KP481    0.33
KP781    0.22
Name: Product, dtype: float64
```

In [72]:

```
round((df["Gender"].value_counts())/len(df),2)
```

Out[72]:

```
Male      0.58
Female    0.42
Name: Gender, dtype: float64
```

In [73]:

```
round((df["MaritalStatus"].value_counts())/len(df),2)
```

Out[73]:

```
Partnered    0.59
Single       0.41
Name: MaritalStatus, dtype: float64
```

In [74]:

```
round(df["Usage"].value_counts()/len(df),2)
```

Out[74]:

```
3    0.38
4    0.29
2    0.18
5    0.09
6    0.04
7    0.01
Name: Usage, dtype: float64
```

In [75]:

```
round(df["Education"].value_counts()/len(df),2)
```

Out[75]:

```
16    0.47
14    0.31
18    0.13
13    0.03
15    0.03
12    0.02
21    0.02
20    0.01
Name: Education, dtype: float64
```

In [77]:

```
round(df["Fitness"].value_counts()/len(df),2)
```

Out[77]:

```
3    0.54
5    0.17
2    0.14
4    0.13
1    0.01
Name: Fitness, dtype: float64
```

In [13]:

```
#Summarizing the number of unique values in each column
unique_counts=pd.DataFrame.from_records([(col,df[col].nunique()) for col in df.columns],columns=['Column_Name','Num_Unique'])
unique_counts.sort_values(by='Num_Unique')
```

Out[13]:

	Column_Name	Num_Unique
2	Gender	2
4	MaritalStatus	2
0	Product	3
6	Fitness	5
5	Usage	6
3	Education	8
1	Age	32
8	Miles	37
7	Income	62

1. Based on the summary of unique values, all the columns except Age, Income and Miles can be considered as categorical

2. Majority customers have total education of 16 to 17 years

3. There is a probability of 0.67 that the customer will use the product 3 to 4 times per week

4. 54% of customers have given themselves a fitness rating of 3

In [14]:

```
#Converting columns to category datatype to reduce memory usage
cols_exc=['Age','Miles','Income']
for i in df.columns:
    if i not in cols_exc:
        df[i]=df[i].astype('category')
```

In [15]:

```
#Checking the reduction in memory usage
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    category
 1   Age            180 non-null    int64   
 2   Gender         180 non-null    category
 3   Education      180 non-null    category
 4   MaritalStatus  180 non-null    category
 5   Usage          180 non-null    category
 6   Fitness        180 non-null    category
 7   Income         180 non-null    int64   
 8   Miles          180 non-null    int64   
dtypes: category(6), int64(3)
memory usage: 6.5 KB
```

3. Visual Analysis

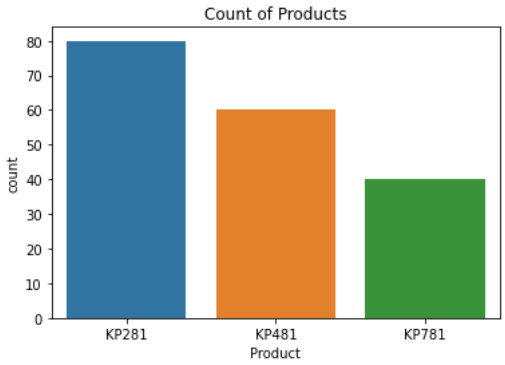
3.1 Univariate Analysis

In [79]:

```
sns.countplot(data=df, x=df["Product"])
plt.title('Count of Products')
```

Out[79]:

```
Text(0.5, 1.0, 'Count of Products')
```

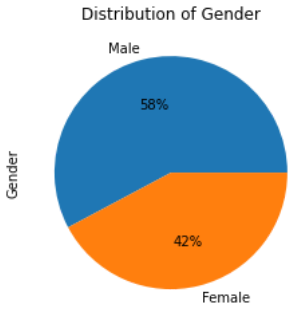


In [80]:

```
df["Gender"].value_counts().plot(kind='pie',autopct='%2.0f%%')
plt.title('Distribution of Gender')
```

Out[80]:

```
Text(0.5, 1.0, 'Distribution of Gender')
```

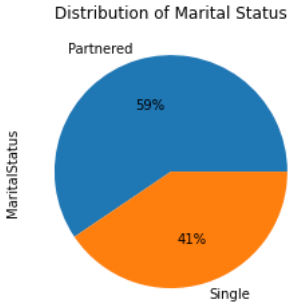


In [81]:

```
df["MaritalStatus"].value_counts().plot(kind='pie',autopct='%2.0f%%')
plt.title('Distribution of Marital Status')
```

Out[81]:

```
Text(0.5, 1.0, 'Distribution of Marital Status')
```

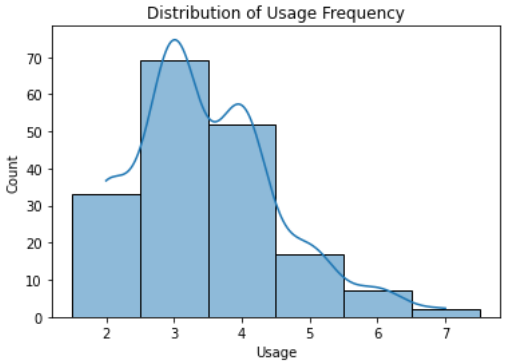


In [83]:

```
sns.histplot(data=df, x=df["Usage"],kde=True)
plt.title('Distribution of Usage Frequency')
```

Out[83]:

```
Text(0.5, 1.0, 'Distribution of Usage Frequency')
```

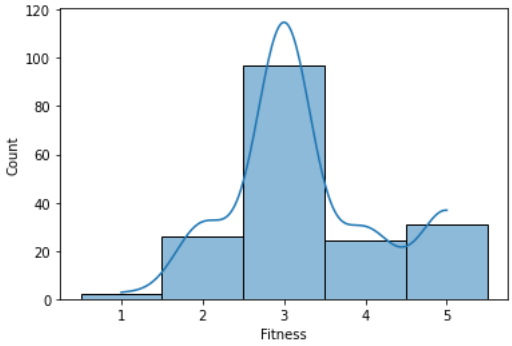


In [23]:

```
sns.histplot(data=df, x=df["Fitness"],kde=True)
plt.title('Distribution of Fitness levels')
```

Out[23]:

```
<AxesSubplot:xlabel='Fitness', ylabel='Count'>
```

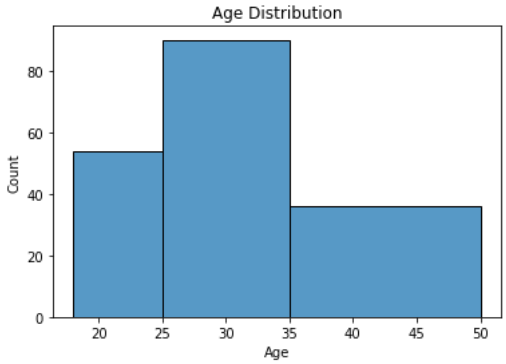


In [84]:

```
sns.histplot(data=df, x=df["Age"],bins=[18,25,35,50])
plt.title('Age Distribution')
```

Out[84]:

```
Text(0.5, 1.0, 'Age Distribution')
```

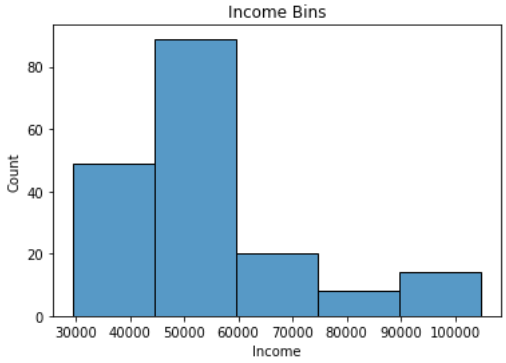


In [105]:

```
sns.histplot(data=df,x=df['Income'],bins=5)
plt.title('Income Bins')
```

Out[105]:

```
Text(0.5, 1.0, 'Income Bins')
```

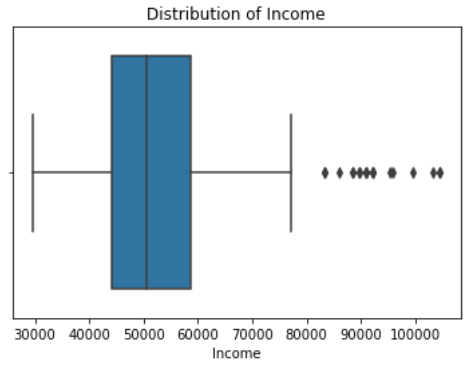


In [85]:

```
sns.boxplot(data=df, x=df["Income"])
plt.title('Distribution of Income')
```

Out[85]:

```
Text(0.5, 1.0, 'Distribution of Income')
```

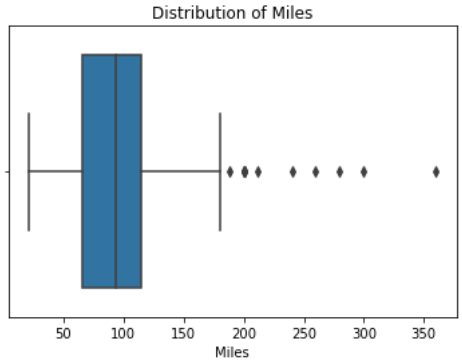


In [86]:

```
sns.boxplot(data=df, x=df["Miles"])
plt.title('Distribution of Miles')
```

Out[86]:

```
Text(0.5, 1.0, 'Distribution of Miles')
```



1. KP281 had the most sales in the last 3 months followed by KP 481 and KP 781

2. Nearly 58% of the customers are Men

3. 59% of the customers are Partnered

4. Most customers fall in the age bracket of 25-35

5. There are more customers with an income between 45000-60000

3.2 Bivariate Analysis

Finding conditional probabilities

In [27]:

```
#Gender vs Product Type
pd.crosstab(index=df["Product"],columns=df["Gender"],margins=True)
```

Out[27]:

Gender	Female	Male	All
Product			
KP281	40	40	80
KP481	29	31	60
KP781	7	33	40
All	76	104	180

P[KP281 | F] = 0.53 and P[KP281 | M]= 0.38

P[KP481 | F] = 0.38 and P[KP481 | M]= 0.30

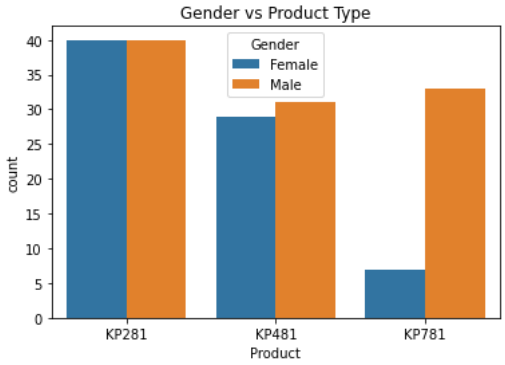
P[KP781 | F] = 0.09 and P[KP781 | M]= 0.32

In [87]:

```
#Gender vs Product Type
sns.countplot(data=df, x=df["Product"],hue=df["Gender"])
plt.title('Gender vs Product Type')
```

Out[87]:

```
Text(0.5, 1.0, 'Gender vs Product Type')
```

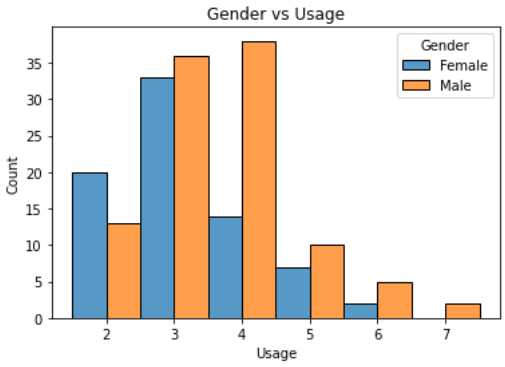


In [88]:

```
# Gender vs Usage
sns.histplot(data=df, x=df["Usage"],hue=df["Gender"],multiple='dodge')
plt.title('Gender vs Usage')
```

Out[88]:

```
Text(0.5, 1.0, 'Gender vs Usage')
```

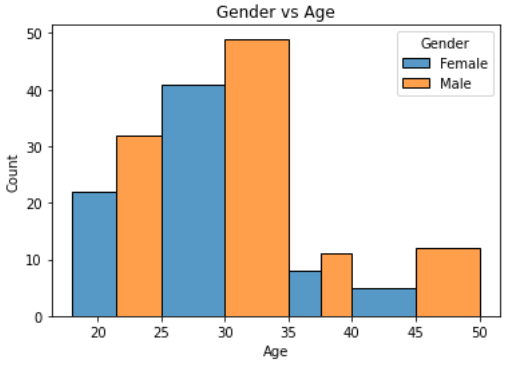


In [89]:

```
# Gender vs Age
sns.histplot(data=df, x=df["Age"],bins=[18,25,35,40,50],hue=df["Gender"],multiple='dodge')
plt.title('Gender vs Age')
```

Out[89]:

```
Text(0.5, 1.0, 'Gender vs Age')
```

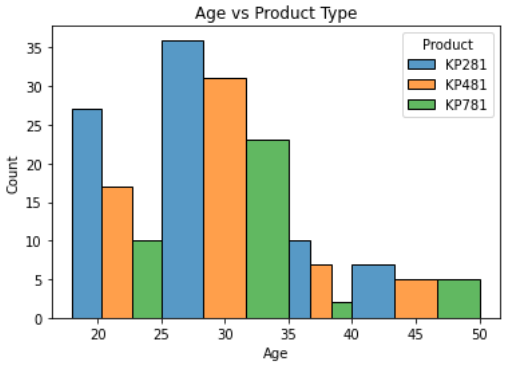


In [90]:

```
#Age vs Product Type
sns.histplot(data=df, x=df["Age"],bins=[18,25,35,40,50],hue=df["Product"],multiple='dodge')
plt.title('Age vs Product Type')
```

Out[90]:

```
Text(0.5, 1.0, 'Age vs Product Type')
```

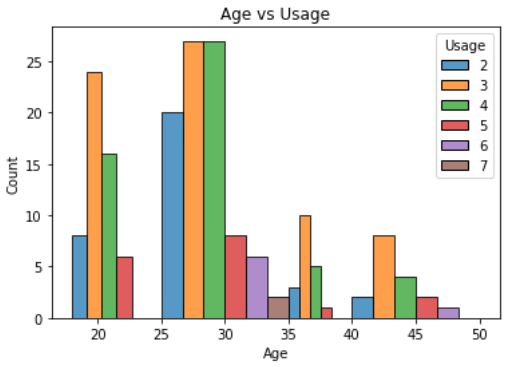


In [91]:

```
# Age vs Usage
sns.histplot(data=df, x=df["Age"],bins=[18,25,35,40,50],hue=df["Usage"],multiple='dodge')
plt.title('Age vs Usage')
```

Out[91]:

```
Text(0.5, 1.0, 'Age vs Usage')
```



In [30]:

```
#Marital Status vs Product Type
pd.crosstab(index=df["Product"],columns=df["MaritalStatus"],margins=True)
```

Out[30]:

MaritalStatus	Partnered	Single	All
Product			
KP281	48	32	80
KP481	36	24	60
KP781	23	17	40
All	107	73	180

P[KP281 | Partnered] = 0.45 and P[KP281 | Single]= 0.44

P[KP481 | Partnered] = 0.34 and P[KP481 | Single]= 0.33

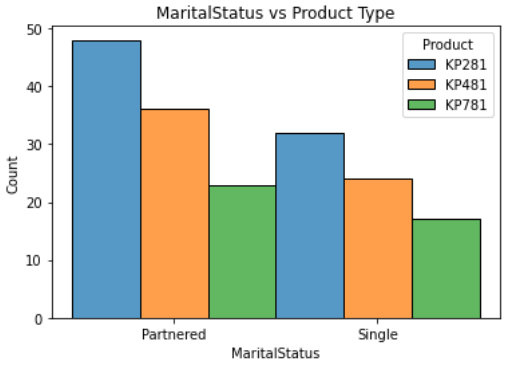
P[KP481 | Partnered] = 0.21 and P[KP781 | Single]= 0.23

In [92]:

```
# Marital Status vs Product Type
sns.histplot(data=df, x=df["MaritalStatus"],hue=df["Product"],multiple='dodge')
plt.title('MaritalStatus vs Product Type')
```

Out[92]:

```
Text(0.5, 1.0, 'MaritalStatus vs Product Type')
```

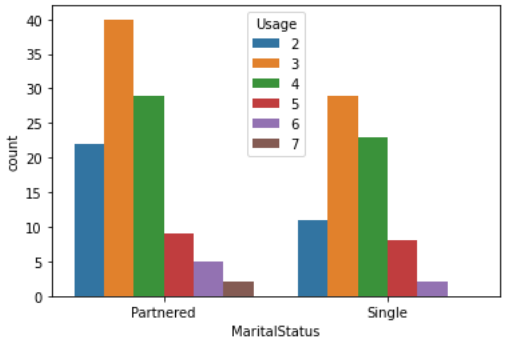


In [68]:

```
# Marital Status vs Usage
sns.countplot(data=df,x=df['MaritalStatus'],hue=df['Usage'])
```

Out[68]:

```
<AxesSubplot:xlabel='MaritalStatus', ylabel='count'>
```



In [32]:

```
#Usage vs Product Type
pd.crosstab(index=df["Product"],columns=df["Usage"],margins=True)
```

Out[32]:

Usage	2	3	4	5	6	7	All
Product							
KP281	19	37	22	2	0	0	80
KP481	14	31	12	3	0	0	60
KP781	0	1	18	12	7	2	40
All	33	69	52	17	7	2	180

P[KP 281 | Usage<=4] = 0.50 and P[KP481 | | Usage<=4]= 0.38 and P[KP781 | | Usage<=4]= 0.12

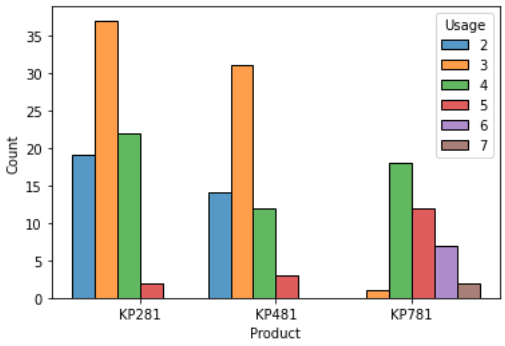
P[KP 281 | Usage>=5] = 0.07 and P[KP481 | | Usage>=5]= 0.12 and P[KP781 | | Usage>=5]= 0.81

In [33]:

```
#Usage vs Product Type
sns.histplot(data=df, x=df["Product"],hue=df["Usage"],multiple='dodge')
```

Out[33]:

```
<AxesSubplot:xlabel='Product', ylabel='Count'>
```



In [34]:

```
#Fitness vs Product Type
pd.crosstab(index=df["Product"],columns=df["Fitness"],margins=True)
```

Out[34]:

Fitness	1	2	3	4	5	All
Product						
KP281	1	14	54	9	2	80
KP481	1	12	39	8	0	60
KP781	0	0	4	7	29	40
All	2	26	97	24	31	180

P[KP281 | Fitness<=3] = 0.55 and P[KP481 | Fitness<=3] = 0.42 and P[KP781 | Fitness<=3] = 0.03

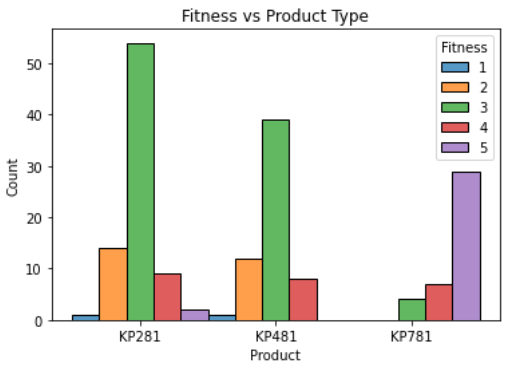
P[KP281 | Fitness>=4] = 0.20 and P[KP481 | Fitness>=4] = 0.15 and P[KP781 | Fitness>=4] = 0.65

In [93]:

```
# Fitness vs Product Type
sns.histplot(data=df, x=df["Product"],hue=df["Fitness"],multiple='dodge')
plt.title('Fitness vs Product Type')
```

Out[93]:

```
Text(0.5, 1.0, 'Fitness vs Product Type')
```

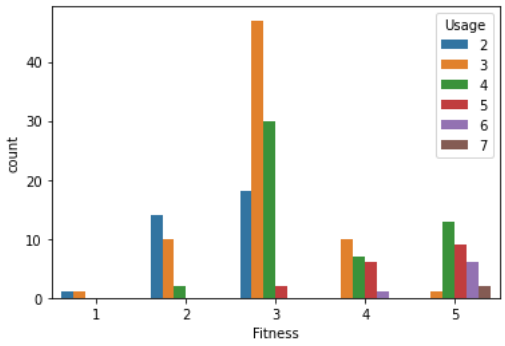


In [58]:

```
# Fitness vs Usage
sns.countplot(data=df,x=df['Fitness'],hue=df['Usage'])
```

Out[58]:

```
<AxesSubplot:xlabel='Fitness', ylabel='count'>
```

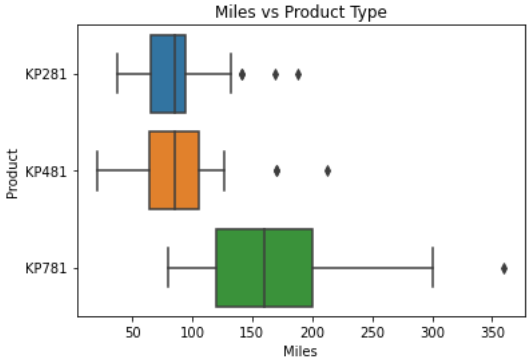


In [94]:

```
# Miles vs Product Type
sns.boxplot(data=df, x=df["Miles"],y=df["Product"])
plt.title('Miles vs Product Type')
```

Out[94]:

```
Text(0.5, 1.0, 'Miles vs Product Type')
```

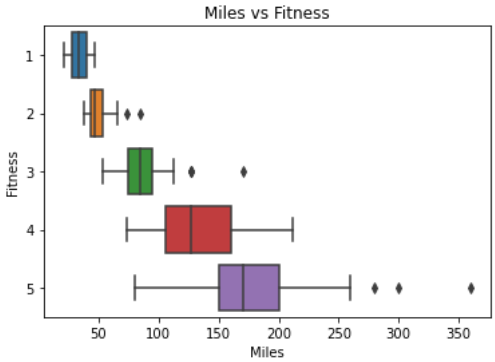


In [95]:

```
# Miles vs Fitness
sns.boxplot(data=df, x=df["Miles"],y=df["Fitness"])
plt.title('Miles vs Fitness')
```

Out[95]:

```
Text(0.5, 1.0, 'Miles vs Fitness')
```

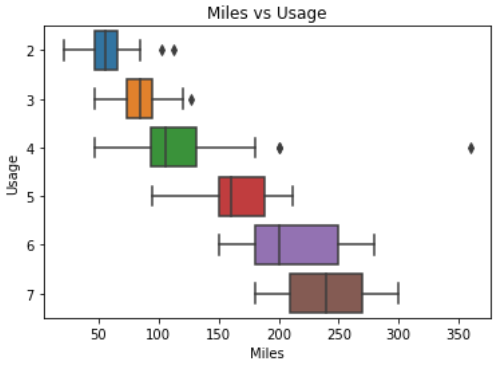


In [96]:

```
# Miles vs Usage
sns.boxplot(data=df, x=df["Miles"],y=df["Usage"])
plt.title('Miles vs Usage')
```

Out[96]:

```
Text(0.5, 1.0, 'Miles vs Usage')
```

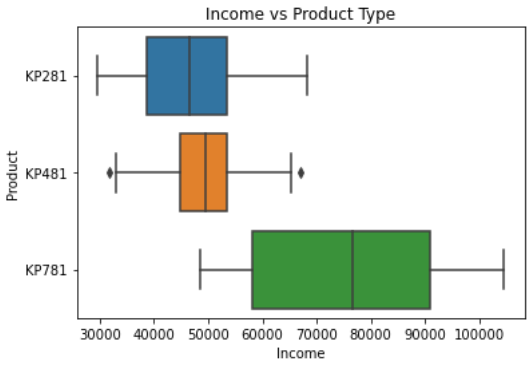


In [97]:

```
#Income vs Product Type
sns.boxplot(data=df, x=df["Income"],y=df["Product"])
plt.title('Income vs Product Type')
```

Out[97]:

```
Text(0.5, 1.0, 'Income vs Product Type')
```

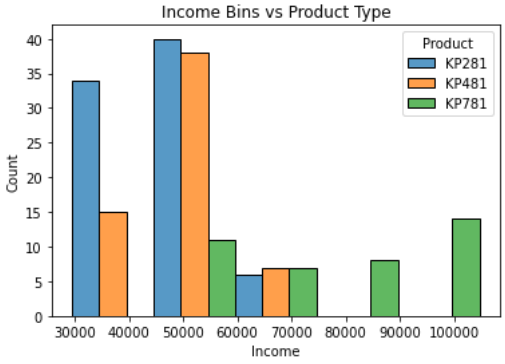


In [108]:

```
# Income Bins vs Product Type
sns.histplot(data=df, x=df["Income"],hue=df['Product'],bins=5,multiple='dodge')
plt.title('Income Bins vs Product Type')
```

Out[108]:

```
Text(0.5, 1.0, 'Income Bins vs Product Type')
```

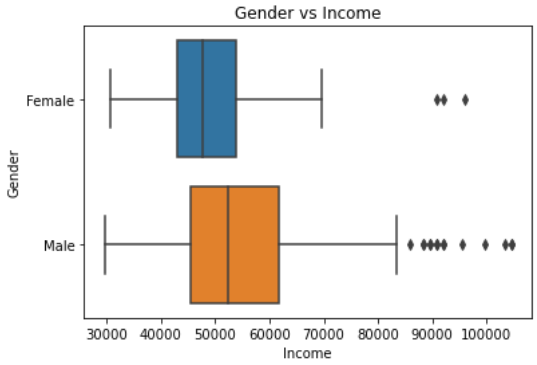


In [98]:

```
# Gender vs Income
sns.boxplot(data=df, x=df["Income"],y=df["Gender"])
plt.title('Gender vs Income')
```

Out[98]:

```
Text(0.5, 1.0, 'Gender vs Income')
```

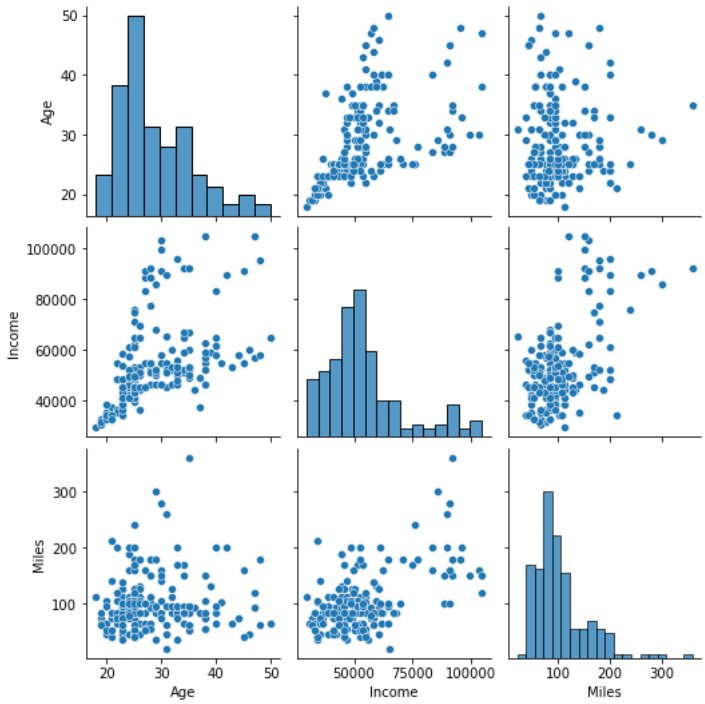


In [100]:

```
# Find correlation between various variables
sns.pairplot(data=df)
```

Out[100]:

```
<seaborn.axisgrid.PairGrid at 0x2a8bc02de50>
```



In [78]:

```
df.corr()
```

Out[78]:

	Age	Income	Miles
Age	1.000000	0.513414	0.036618
Income	0.513414	1.000000	0.543473
Miles	0.036618	0.543473	1.000000

1. The probability of a female buying KP 281 is higher than the other 2 models. KP 781 is least preferred by females. Males

have an almost equal probability of buying all 3 models

2. Age is not a deciding factor in pitching a model to a customer as all age groups have similar preferences

3. Marital Status is also not a deciding factor in understanding customer preferences

4. The customers who consider themselves more fit (>4) are more likely to exercise 5 or more days in a week and target to cover more miles. They are more likely to prefer the KP 781 model over the others. The customers who have given a fitness rating of 3 and less also want to exercise less number of days with less number of miles per week and prefer the KP281 and KP 481 models

5. Apart from fitness and usage, income also influences the customers' decision. KP 781 being more expensive is preferred by those with a higher income, whereas KP 281 and KP 481 are the preferred models for those on a budget.

6. There is a moderately positive correlation between Age and Income and also between Income and Miles. The customers in the higher age group have more buying capacity and also plan to run more miles. They can be ideal customers for the KP 781 model.

4. Outlier Detection

In [43]:

```
df.describe()
```

Out[43]:

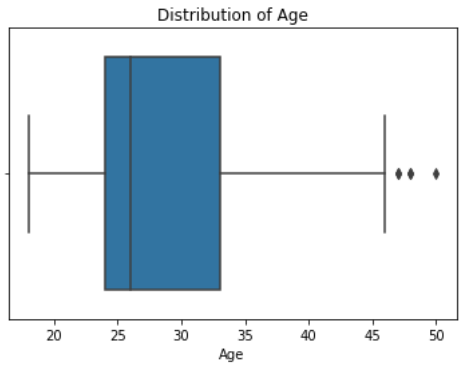
	Age	Income	Miles
count	180.000000	180.000000	180.000000
mean	28.788889	53719.577778	103.194444
std	6.943498	16506.684226	51.863605
min	18.000000	29562.000000	21.000000
25%	24.000000	44058.750000	66.000000
50%	26.000000	50596.500000	94.000000
75%	33.000000	58668.000000	114.750000
max	50.000000	104581.000000	360.000000

In [101]:

```
# Visualize outliers using statistical methods
# Age Column
sns.boxplot(data=df,x=df["Age"])
plt.title('Distribution of Age')
```

Out[101]:

```
Text(0.5, 1.0, 'Distribution of Age')
```



In [45]:

```
# Detect outlier values using statistical methods
q1=np.percentile(df["Age"],75)
q2=np.percentile(df["Age"],25)
IQR=q1-q2
outliers =df[(df["Age"]<(q2-1.5*IQR)) | (df["Age"]>(q1 + 1.5*IQR))]["Age"]
outliers.value_counts()
```

Out[45]:

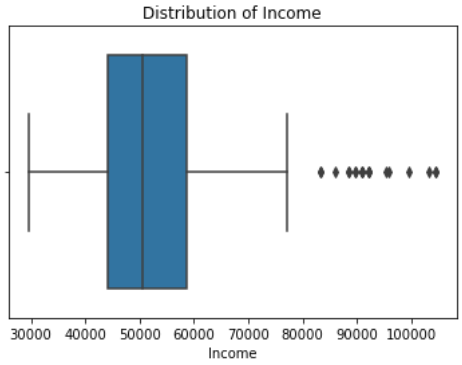
```
47    2
48    2
50    1
Name: Age, dtype: int64
```

In [102]:

```
# Income Column
sns.boxplot(data=df,x=df["Income"])
plt.title('Distribution of Income')
```

Out[102]:

```
Text(0.5, 1.0, 'Distribution of Income')
```



In [47]:

```
# Detect outlier values using statistical methods
q1=np.percentile(df["Income"],75)
q2=np.percentile(df["Income"],25)
IQR=q1-q2
outliers =df[(df["Income"]<(q2-1.5*IQR)) | (df["Income"]>(q1 + 1.5*IQR))]["Income"]
outliers.value_counts()
```

Out[47]:

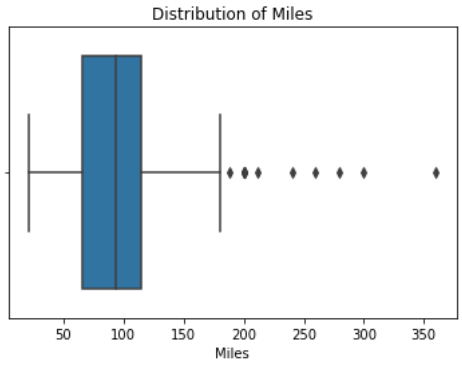
```
90886     3
92131     3
83416     2
88396     2
89641     2
104581    2
85906     1
103336    1
99601     1
95866     1
95508     1
Name: Income, dtype: int64
```

In [103]:

```
# Miles Column
sns.boxplot(data=df,x=df["Miles"])
plt.title('Distribution of Miles')
```

Out[103]:

```
Text(0.5, 1.0, 'Distribution of Miles')
```



In [49]:

```
# Detect outlier values using statistical methods
q1=np.percentile(df["Miles"],75)
q2=np.percentile(df["Miles"],25)
IQR=q1-q2
outliers =df[(df["Miles"]<(q2-1.5*IQR)) | (df["Miles"]>(q1 + 1.5*IQR))]["Miles"]
outliers.value_counts()
```

Out[49]:

```
200    6
188    1
212    1
240    1
300    1
280    1
260    1
360    1
Name: Miles, dtype: int64
```