# Aerofit

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.

# Dataset Description

The company collected the data on individuals who purchased a treadmill from the AeroFit stores during the prior three months. The dataset has the following features:

Product Purchased: KP281, KP481, or KP781

Age: In years

Gender: Male/Female

Education: In years

MaritalStatus: Single or partnered

Usage: The average number of times the customer plans to use the treadmill each week.

Income: Annual income (in $)

Fitness: Self-rated fitness on a 1-to-5 scale, where 1 is the poor shape and 5 is the excellent shape.

Miles: The average number of miles the customer expects to walk/run each week Product Portfolio:

The KP281 is an entry-level treadmill that sells for $1,500. The KP481 is for mid-level runners that sell for 1,750. The KP781 treadmill is having advanced features that sell for $2,500.

In [202…

```
import pandas as pd
import numpy as np
```

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

In [203…
```
!wget d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/125/original/aerofit_treadmill.csv
```

```
--2024-04-12 16:18:44--  http://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/125/original/aerofit_treadmill.csv
Resolving d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)... 18.172.139.46, 18.172.139.210, 18.172.139.61, ...
Connecting to d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)|18.172.139.46|:80... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/125/original/aerofit_treadmill.csv [following]
--2024-04-12 16:18:44--  https://d2beiqkhq929f0.cloudfront.net/public_assets/assets/000/001/125/original/aerofit_treadmill.csv
Connecting to d2beiqkhq929f0.cloudfront.net (d2beiqkhq929f0.cloudfront.net)|18.172.139.46|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7279 (7.1K) [text/plain]
Saving to: ‘aerofit_treadmill.csv.2’

aerofit_treadmill.c 100%[===================>]   7.11K  --.-KB/s    in 0s

2024-04-12 16:18:44 (1.76 GB/s) - ‘aerofit_treadmill.csv.2’ saved [7279/7279]
```

In [204…
```
#Loading data
aerofit = pd.read_csv("aerofit_treadmill.csv")
```

In [205…
```
aerofit.head()
```

Out[205]:

| | 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 [206…
```
#Shape
aerofit.shape
```

Out[206]:
```
(180, 9)
```

In [207…
```
aerofit.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 [208…
```
aerofit.duplicated().any()
```

Out[208]:
```
False
```

# Observation

There are 180 rows and 9 columns in the entire dataset. There are no null values or duplicate values in the data.

# Missing values

In [209…
```
aerofit["Miles"].isna().sum()
```

Out[209]:
```
0
```

In [210…
```
aerofit["MaritalStatus"].isna().sum()
```

Out[210]:
```
0
```

In [211…
```
aerofit["Age"].isna().sum()
```
Out[211]:
```
0
```

In [212…
```
aerofit["Product"].isna().sum()
```
Out[212]:
```
0
```

In [213…
```
aerofit["Gender"].isna().sum()
```
Out[213]:
```
0
```

In [214…
```
aerofit["Fitness"].isna().sum()
```
Out[214]:
```
0
```

In [215…
```
aerofit["Education"].isna().sum()
```
Out[215]:
```
0
```

In [216…
```
aerofit["Usage"].isna().sum()
```
Out[216]:
```
0
```

In [217…
```
aerofit["Income"].isna().sum()
```
Out[217]:
```
0
```

# Observation

There are no missing values in the data. The data is completely clean and can we analysed directly.

In [218…
```
aerofit.describe()
```

Out[218]:

| | 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 |

## Descriptive Analysis

- Total count of all columns is 180
- Age: Mean age of the customer is 28 years, half of the customer's mean age is 26.
- Education: Mean Education is 15 with maximum as 21 and minimum as 12.
- Usage: Mean Usage per week is 3.4, with maximum as 7 and minimum as 2.
- Fitness: Average rating is 3.3 on a scale of 1 to 5.
- Miles: Average number of miles the customer walks is 103 with maximum distance travelled by most people is almost 115 and minimum is 21.
- Income (in $): Most customer earns around 58K annually, with maximum of 104K and minimum almost 30K

# Conversion of Numerical columns to Categorical

In [219…

```
# Conversion of fitness rating into categories
def fit_cat(x):
    if x["Fitness"] <=2:
         x["fitness_cat"] = "Bad"
    elif x["Fitness"] >=3 and x["Fitness"] <=4:
         x["fitness_cat"]= "Good"
    else:
```

```
        x["fitness_cat"] ="Best"
    return x
aerofit = aerofit.apply(fit_cat, axis =1)
aerofit
```

Out[219]:

| | Product | Age | Gender | Education | MaritalStatus | Usage | Fitness | Income | Miles | fitness_cat |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | KP281 | 18 | Male | 14 | Single | 3 | 4 | 29562 | 112 | Good |
| **1** | KP281 | 19 | Male | 15 | Single | 2 | 3 | 31836 | 75 | Good |
| **2** | KP281 | 19 | Female | 14 | Partnered | 4 | 3 | 30699 | 66 | Good |
| **3** | KP281 | 19 | Male | 12 | Single | 3 | 3 | 32973 | 85 | Good |
| **4** | KP281 | 20 | Male | 13 | Partnered | 4 | 2 | 35247 | 47 | Bad |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **175** | KP781 | 40 | Male | 21 | Single | 6 | 5 | 83416 | 200 | Best |
| **176** | KP781 | 42 | Male | 18 | Single | 5 | 4 | 89641 | 200 | Good |
| **177** | KP781 | 45 | Male | 16 | Single | 5 | 5 | 90886 | 160 | Best |
| **178** | KP781 | 47 | Male | 18 | Partnered | 4 | 5 | 104581 | 120 | Best |
| **179** | KP781 | 48 | Male | 18 | Partnered | 4 | 5 | 95508 | 180 | Best |

180 rows × 10 columns

In [220…

```
# Conversion of income into categories
def income_cat(x):
    if x["Income"] <=40000:
         x["income_cat"] = "Low"
    elif x["Income"] >=40000 and x["Income"] <=80000:
         x["income_cat"]= "Medium"
    else:
        x["income_cat"] ="High"
    return x
aerofit = aerofit.apply(income_cat, axis =1)
aerofit
```

Out[220]:

| | Product | Age | Gender | Education | MaritalStatus | Usage | Fitness | Income | Miles | fitness_cat | income_cat |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | KP281 | 18 | Male | 14 | Single | 3 | 4 | 29562 | 112 | Good | Low |
| **1** | KP281 | 19 | Male | 15 | Single | 2 | 3 | 31836 | 75 | Good | Low |
| **2** | KP281 | 19 | Female | 14 | Partnered | 4 | 3 | 30699 | 66 | Good | Low |
| **3** | KP281 | 19 | Male | 12 | Single | 3 | 3 | 32973 | 85 | Good | Low |
| **4** | KP281 | 20 | Male | 13 | Partnered | 4 | 2 | 35247 | 47 | Bad | Low |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **175** | KP781 | 40 | Male | 21 | Single | 6 | 5 | 83416 | 200 | Best | High |
| **176** | KP781 | 42 | Male | 18 | Single | 5 | 4 | 89641 | 200 | Good | High |
| **177** | KP781 | 45 | Male | 16 | Single | 5 | 5 | 90886 | 160 | Best | High |
| **178** | KP781 | 47 | Male | 18 | Partnered | 4 | 5 | 104581 | 120 | Best | High |
| **179** | KP781 | 48 | Male | 18 | Partnered | 4 | 5 | 95508 | 180 | Best | High |

180 rows × 11 columns

In [221…

```
# Conversion of age into categories
def age_cat(x):
    if x["Age"] <=25:
         x["age_cat"] = "Young"
    elif x["Age"] >=26 and x["Age"] <=40:
         x["age_cat"]= "Adult"
    else:
        x["age_cat"] ="Old"
    return x
aerofit = aerofit.apply(age_cat, axis =1)
aerofit
```

Out[221]:

| | Product | Age | Gender | Education | MaritalStatus | Usage | Fitness | Income | Miles | fitness_cat | income_cat | age_cat |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | KP281 | 18 | Male | 14 | Single | 3 | 4 | 29562 | 112 | Good | Low | Young |
| **1** | KP281 | 19 | Male | 15 | Single | 2 | 3 | 31836 | 75 | Good | Low | Young |
| **2** | KP281 | 19 | Female | 14 | Partnered | 4 | 3 | 30699 | 66 | Good | Low | Young |
| **3** | KP281 | 19 | Male | 12 | Single | 3 | 3 | 32973 | 85 | Good | Low | Young |
| **4** | KP281 | 20 | Male | 13 | Partnered | 4 | 2 | 35247 | 47 | Bad | Low | Young |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **175** | KP781 | 40 | Male | 21 | Single | 6 | 5 | 83416 | 200 | Best | High | Adult |
| **176** | KP781 | 42 | Male | 18 | Single | 5 | 4 | 89641 | 200 | Good | High | Old |
| **177** | KP781 | 45 | Male | 16 | Single | 5 | 5 | 90886 | 160 | Best | High | Old |
| **178** | KP781 | 47 | Male | 18 | Partnered | 4 | 5 | 104581 | 120 | Best | High | Old |
| **179** | KP781 | 48 | Male | 18 | Partnered | 4 | 5 | 95508 | 180 | Best | High | Old |

180 rows × 12 columns

Few Numerical columns are converted into categorical columns.

# Value Counts

In [222...
```
aerofit["Product"].value_counts()
```

Out[222]:
```
Product
KP281    80
KP481    60
KP781    40
Name: count, dtype: int64
```

In [223...
```
aerofit["Age"].value_counts().head(10)
```

Out[223]:
```
Age
25    25
23    18
24    12
26    12
28     9
35     8
33     8
30     7
38     7
21     7
Name: count, dtype: int64
```

In [224…
```
aerofit["Education"].value_counts()
```

Out[224]:
```
Education
16    85
14    55
18    23
15     5
13     5
12     3
21     3
20     1
Name: count, dtype: int64
```

In [225…
```
aerofit["MaritalStatus"].value_counts()
```

Out[225]:
```
MaritalStatus
Partnered    107
Single        73
Name: count, dtype: int64
```

In [226…
```
aerofit['Usage'].value_counts()
```

Out[226]:
```
Usage
3    69
4    52
2    33
5    17
6     7
7     2
Name: count, dtype: int64
```

In [227…
```
aerofit['Fitness'].value_counts()
```

Out[227]:
```
Fitness
3    97
5    31
2    26
4    24
1     2
Name: count, dtype: int64
```

In [228…
```
aerofit['Miles'].value_counts().head(20)
```

Out[228]:
```
Miles
85     27
95     12
66     10
75     10
47      9
106     9
94      8
113     8
53      7
100     7
180     6
200     6
56      6
64      6
127     5
160     5
42      4
150     4
38      3
74      3
Name: count, dtype: int64
```

In [229…
```
aerofit['Income'].value_counts()
```

Out[229]:
```
Income
45480    14
52302     9
46617     8
54576     8
53439     8
         ..
65220     1
55713     1
68220     1
30699     1
95508     1
Name: count, Length: 62, dtype: int64
```

In [230…
```
aerofit['Fitness'].value_counts()
```

Out[230]:
```
Fitness
3    97
5    31
2    26
4    24
1     2
Name: count, dtype: int64
```

In [231…
```
aerofit['fitness_cat'].value_counts()
```

Out[231]:
```
fitness_cat
Good    121
Best     31
Bad      28
Name: count, dtype: int64
```

In [232…
```
aerofit['income_cat'].value_counts()
```

Out[232]:
```
income_cat
Medium    129
Low        32
High       19
Name: count, dtype: int64
```

In [233…
```
aerofit['Gender'].value_counts()
```

Out[233]:
```
Gender
Male      104
Female     76
Name: count, dtype: int64
```

In [234…
```
aerofit['Income'].unique()
```

Out[234]:
```
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])
```

In [235…
```
aerofit['Miles'].unique()
```

Out[235]:
```
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])
```

In [236…
```
aerofit['Education'].unique()
```

Out[236]:
```
array([14, 15, 12, 13, 16, 18, 20, 21])
```

In [237…
```
aerofit['Product'].unique()
```

Out[237]:
```
array(['KP281', 'KP481', 'KP781'], dtype=object)
```

In [238…
```
aerofit['MaritalStatus'].unique()
```

Out[238]:
```
array(['Single', 'Partnered'], dtype=object)
```

In [239…
```
aerofit['Usage'].unique()
```

Out[239]:
```
array([3, 2, 4, 5, 6, 7])
```

In [240…
```
aerofit["Age"].nunique()
```

Out[240]:
```
32
```

# Observation

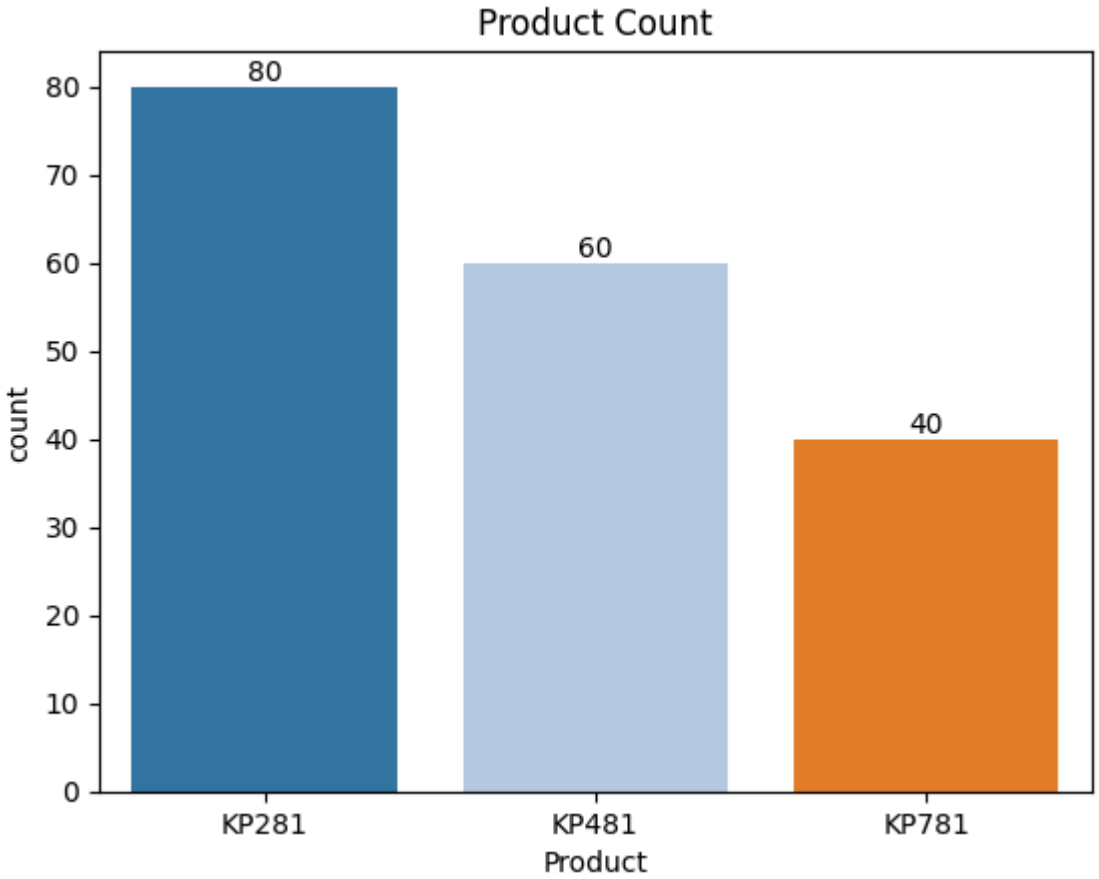
- KP281, KP481, KP781 are the 3 different products
- Most commonly purchased treadmill product type is KP281
- There are 32 unique ages
- 104 Males and 76 Females are in the customers list
- 8 unique set of Educations (14, 15, 12, 13, 16, 18, 20, 21)
- Highest rated Fitness rating is 3
- Most customers usage treadmill atleast 3 days per week
- Majority of the customers who have purchased are Married/Partnered

# Visual Analytics

In [241…

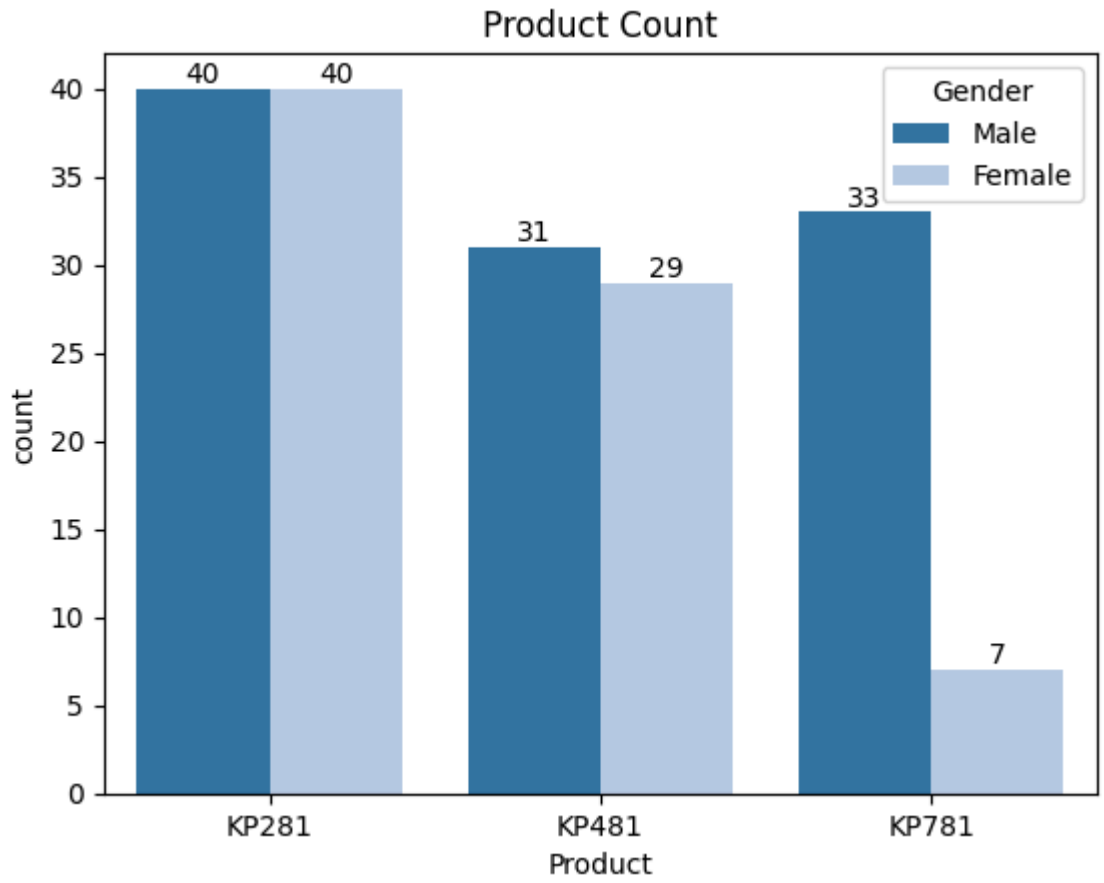
```
product_count = sns.countplot(data = aerofit, x = "Product", palette= "tab20")
for i in product_count.containers:
     product_count.bar_label(i)
plt.title("Product Count")
plt.show()
```

```
<ipython-input-241-0dfee7d5f38b>:1: 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 effect.

  product_count = sns.countplot(data = aerofit, x = "Product", palette= "tab20")
```




In [242…

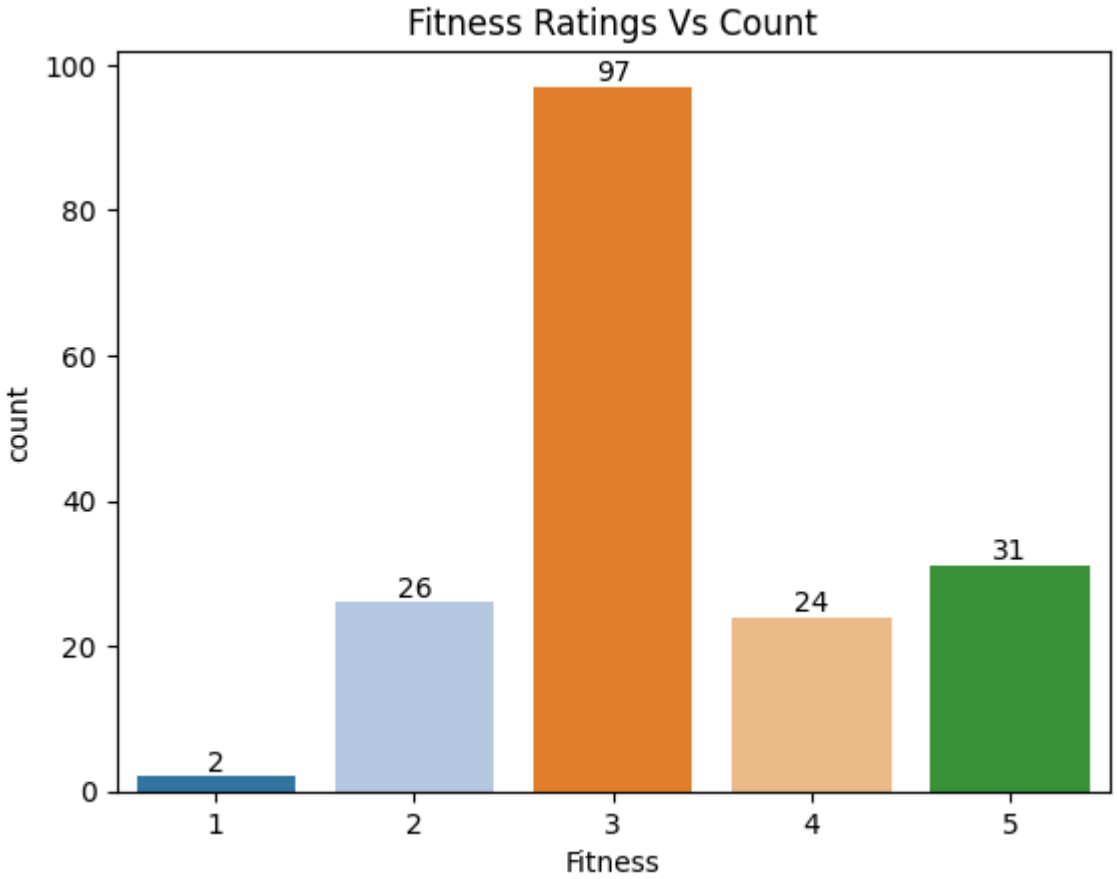
```
gender_per_product = sns.countplot(data = aerofit, x = "Product", hue = "Gender", palette= "tab20")
for i in gender_per_product.containers:
     gender_per_product.bar_label(i)
plt.title("Product Count")
plt.show()
```




In [243…

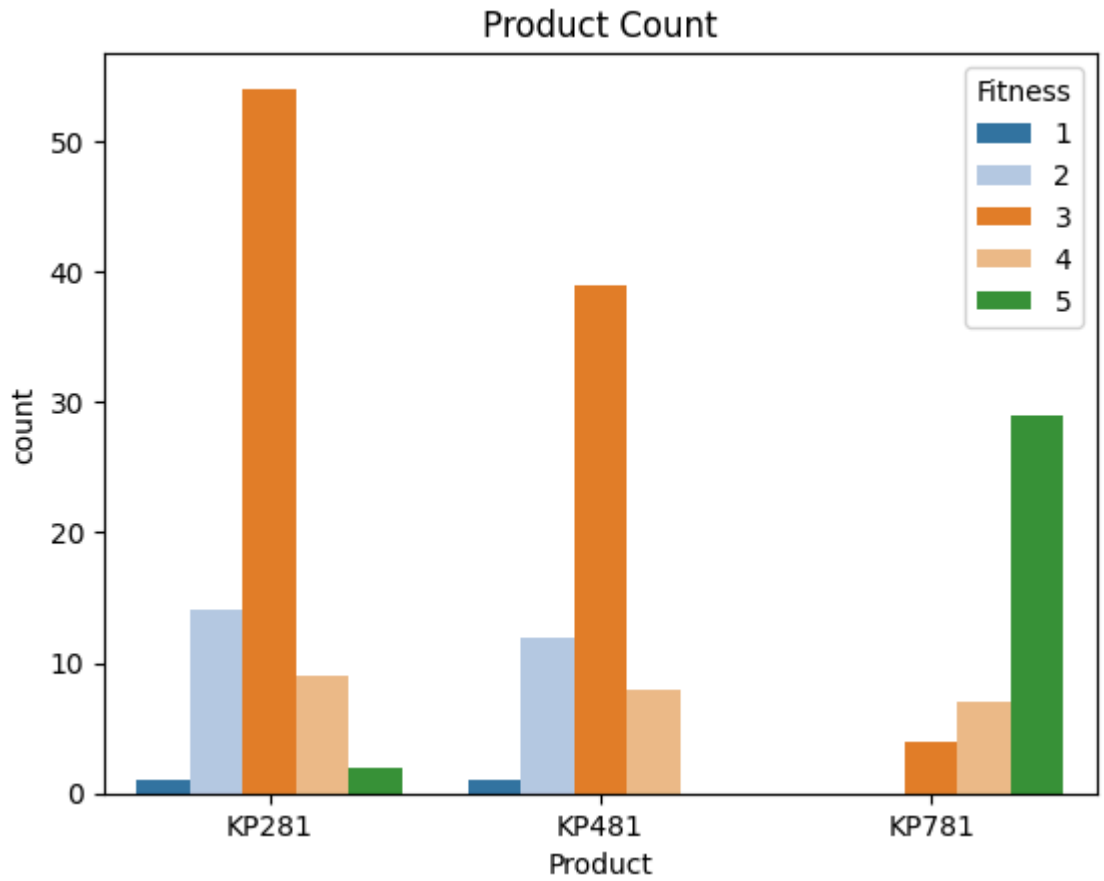
```
fitness_count = sns.countplot(data = aerofit, x= "Fitness", palette = "tab20")
for i in fitness_count.containers:
    fitness_count.bar_label(i)
plt.title("Fitness Ratings Vs Count")
plt.show()
```

```
<ipython-input-243-72d58d3ad1a8>:1: 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 effect.

  fitness_count = sns.countplot(data = aerofit, x= "Fitness", palette = "tab20")
```



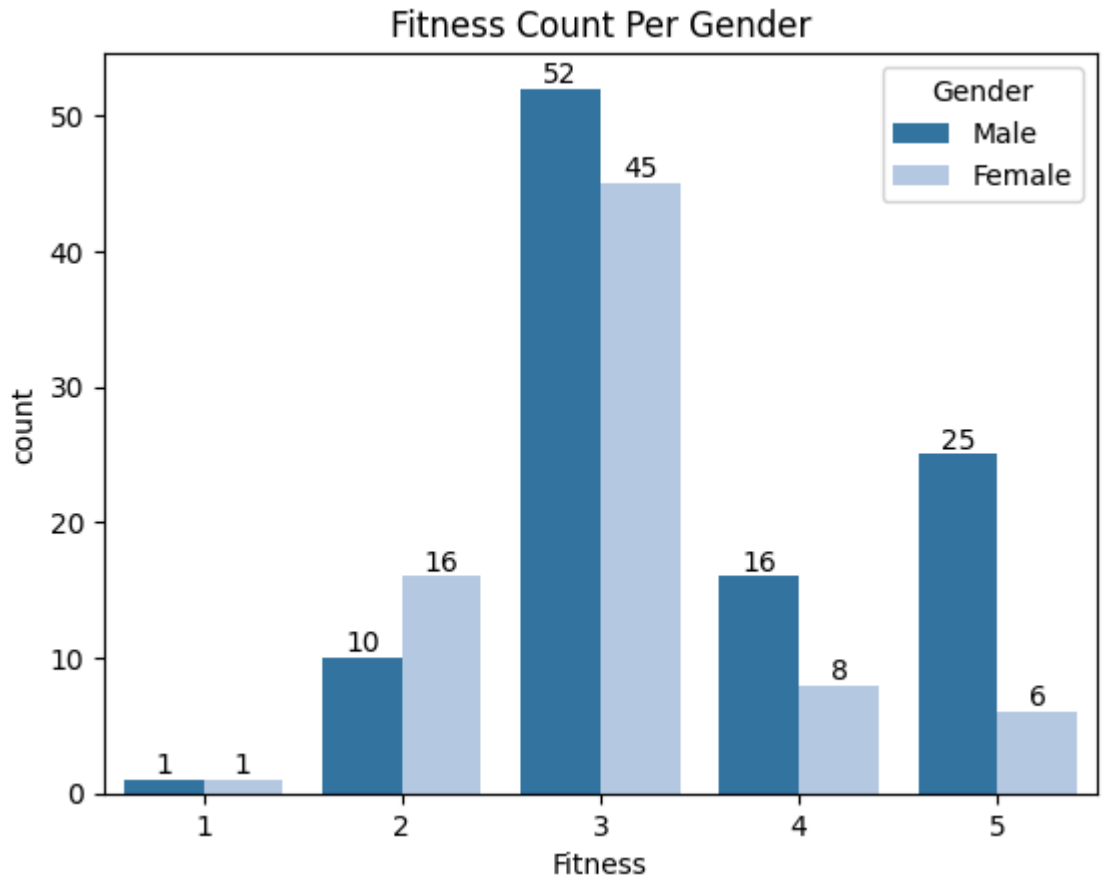

In [244…

```
sns.countplot(data = aerofit, x = "Product", hue = "Fitness", palette= "tab20")
plt.title("Product Count")
plt.show()
```



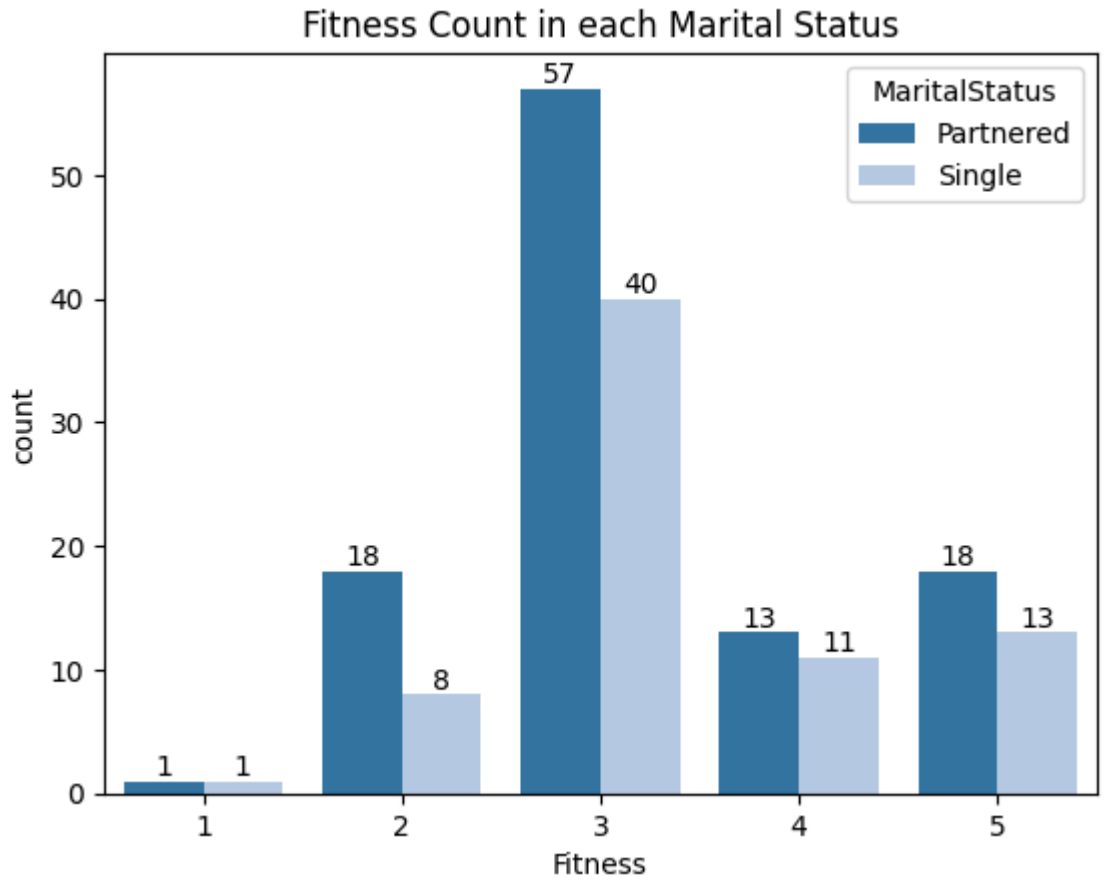

In [245…
```
fitness_count_per_gender = sns.countplot(data = aerofit, x= 'Fitness', hue = "Gender", palette = "tab20")
for i in fitness_count_per_gender.containers:
    fitness_count_per_gender.bar_label(i)
plt.title("Fitness Count Per Gender")
```

Out[245]:
```
Text(0.5, 1.0, 'Fitness Count Per Gender')
```




In [246…
```
fitness_count = sns.countplot(data = aerofit, x= 'Fitness', hue = "MaritalStatus", palette = "tab20")
for i in fitness_count.containers:
    fitness_count.bar_label(i)
plt.title("Fitness Count in each Marital Status")
```

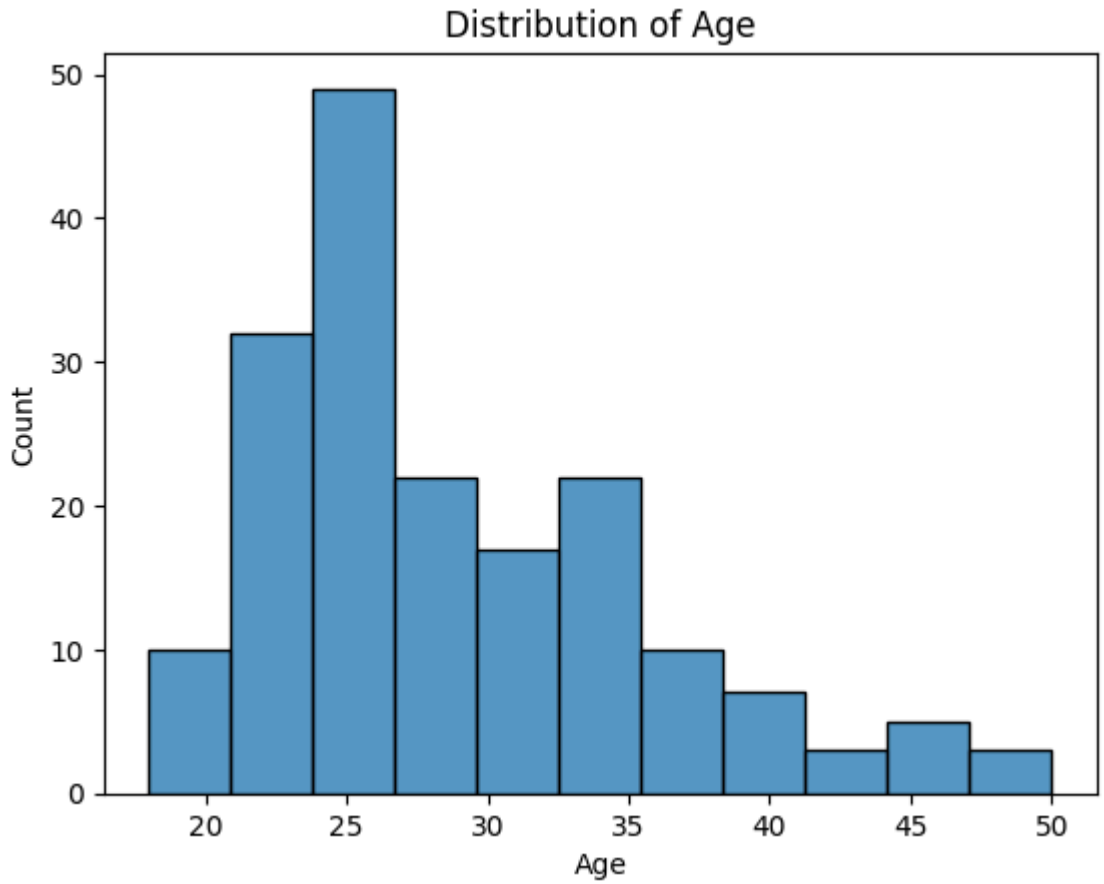
Out[246]:
```
Text(0.5, 1.0, 'Fitness Count in each Marital Status')
```




In [247…

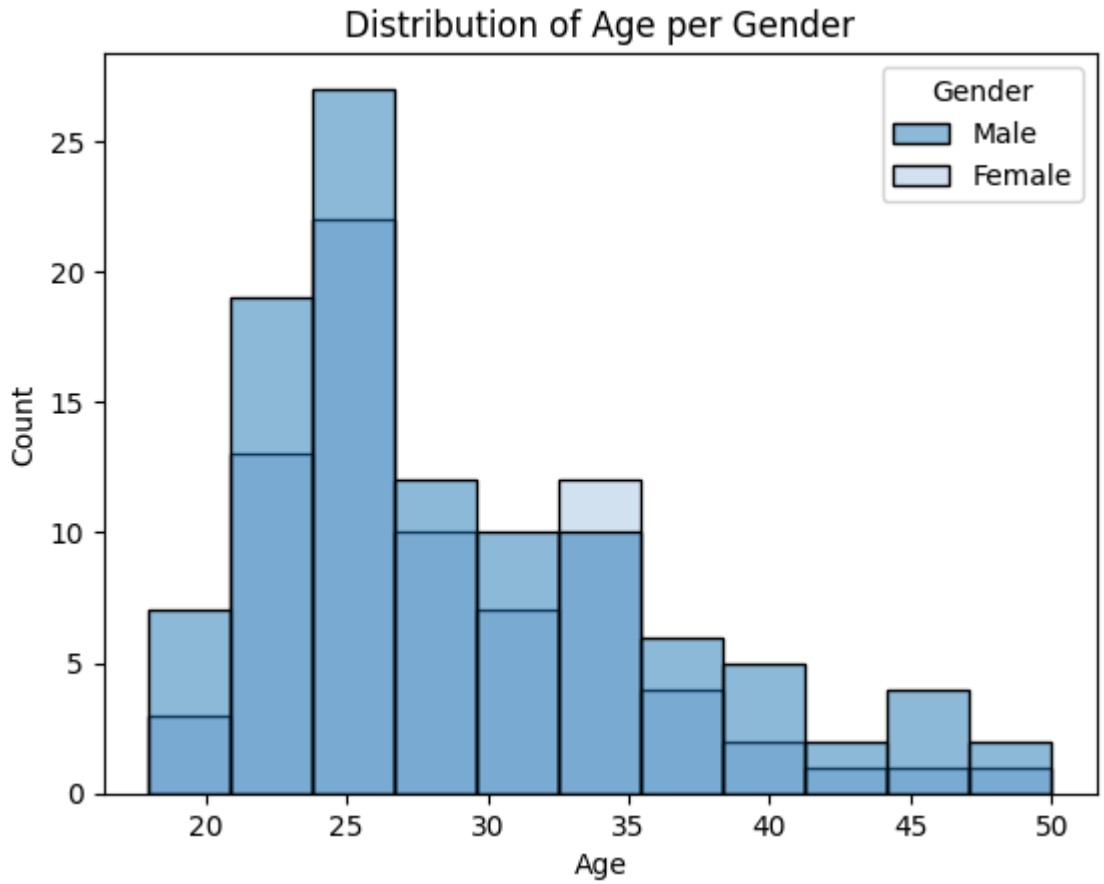
```
sns.histplot(aerofit["Age"], palette = "tab20")
plt.title("Distribution of Age")
plt.show()
```

```
<ipython-input-247-56b3cf55a49d>:1: UserWarning: Ignoring `palette` because no `hue` variable has been assigned.
  sns.histplot(aerofit["Age"], palette = "tab20")
```



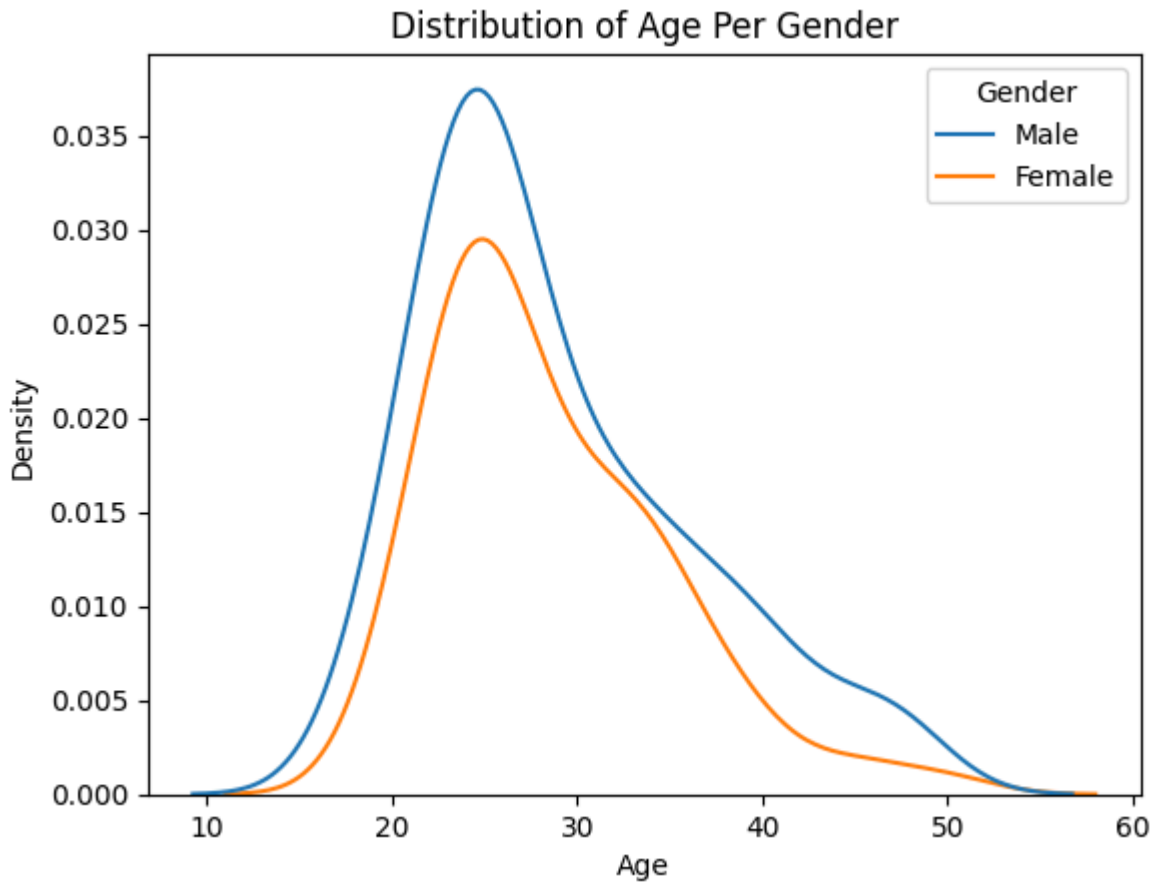

In [248…

```
sns.histplot(data = aerofit, x= "Age", hue = "Gender", palette = "tab20")
plt.title("Distribution of Age per Gender")
plt.show()
```




In [249…

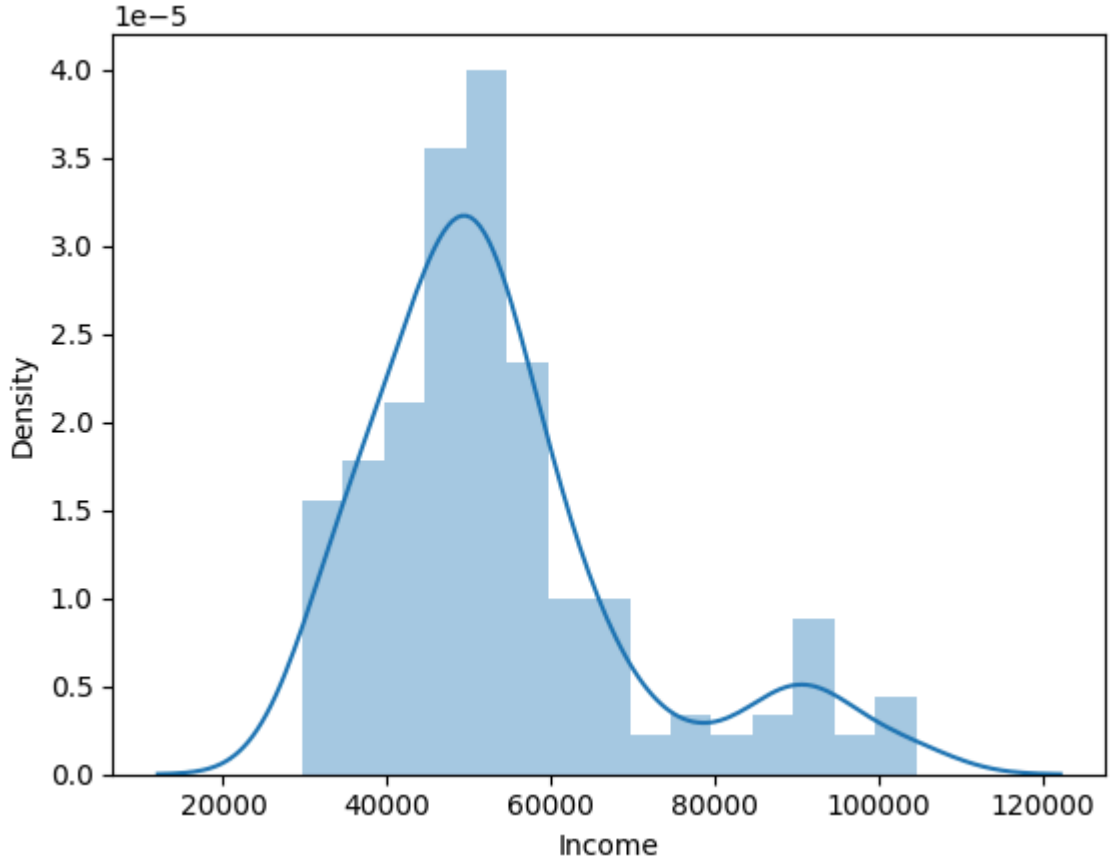
```
sns.kdeplot(data = aerofit, x="Age", hue = "Gender")
plt.title("Distribution of Age Per Gender")
plt.show()
```




In [250…

```
sns.distplot(aerofit["Income"], kde = True)
plt.show()
```

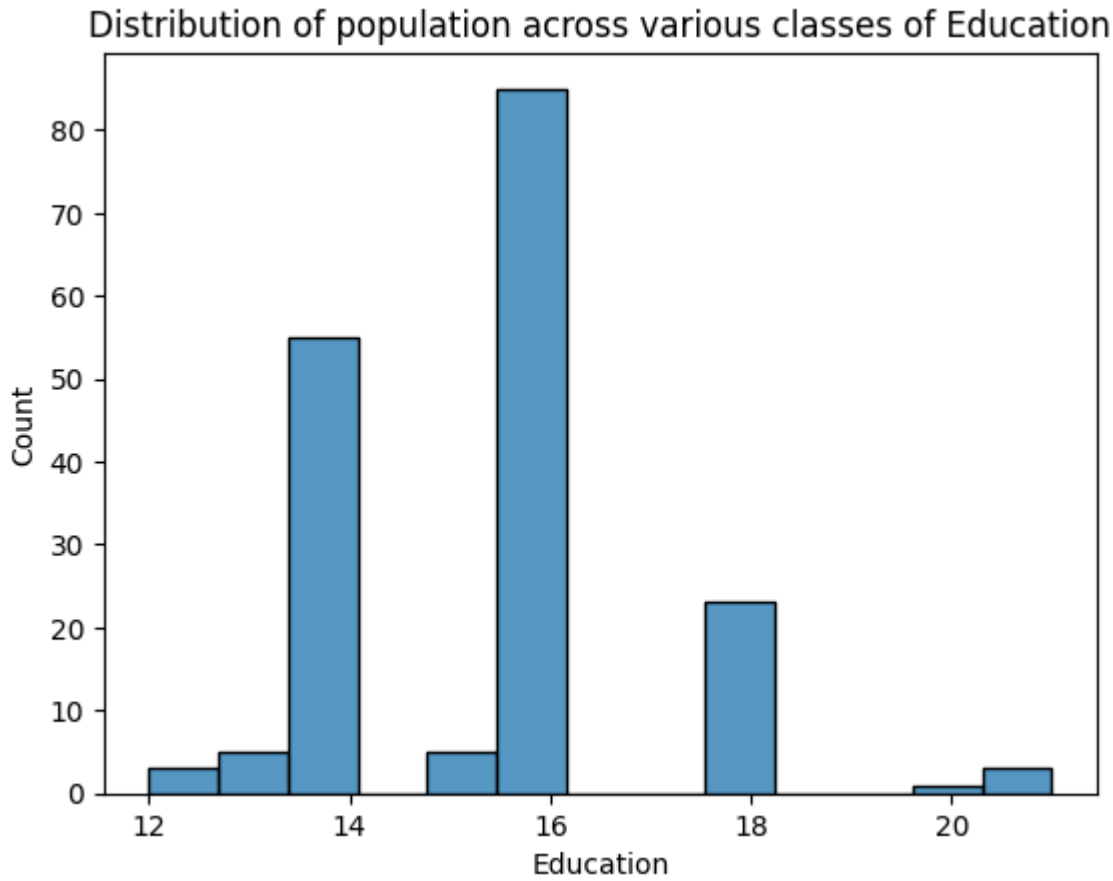
```
<ipython-input-250-89054bcecfb2>:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(aerofit["Income"], kde = True)
```




In [251…
```
# Education Analysis - Histogram
sns.histplot(data=aerofit,x='Education', palette = "tab20")
plt.title("Distribution of population across various classes of Education")
```

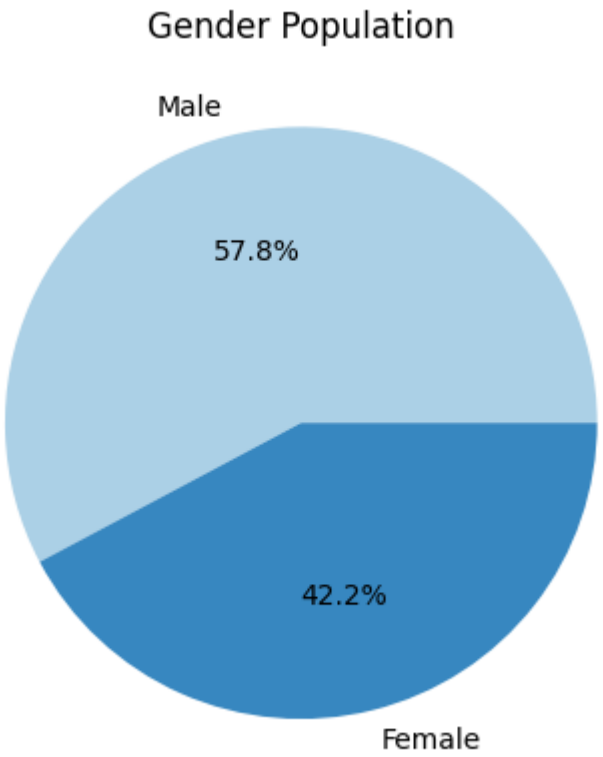
```
<ipython-input-251-686c59f926df>:2: UserWarning: Ignoring `palette` because no `hue` variable has been assigned.
  sns.histplot(data=aerofit,x='Education', palette = "tab20")
```

Out[251]:
```
Text(0.5, 1.0, 'Distribution of population across various classes of Education')
```



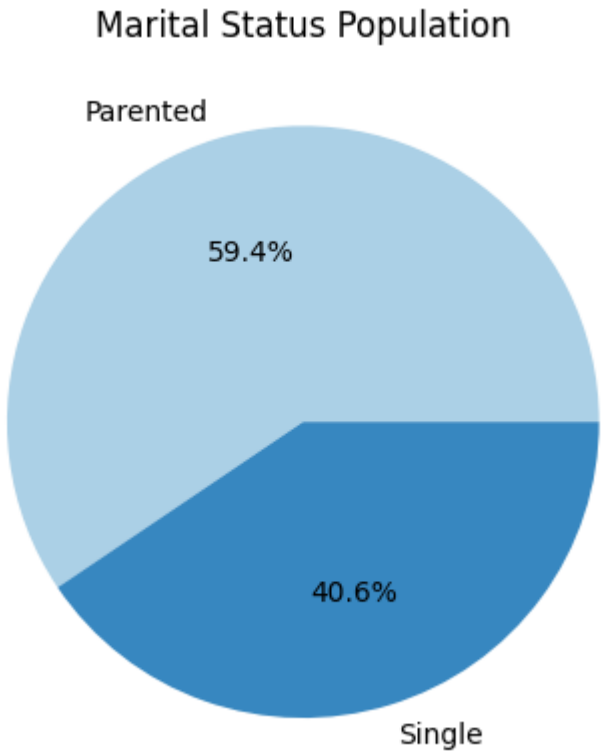

In [252…

```
colors = sns.color_palette('Blues', 2)
plt.pie(aerofit["Gender"].value_counts(), labels = (["Male", "Female"]), autopct='%1.1f%%', colors = colors)
plt.title("Gender Population")
plt.show()
```




In [253…

```
colors = sns.color_palette('Blues', 2)
plt.pie(aerofit["MaritalStatus"].value_counts(), labels = (["Parented", "Single"]), autopct='%1.1f%%', colors = colors)
plt.title("Marital Status Population")
plt.show()
```



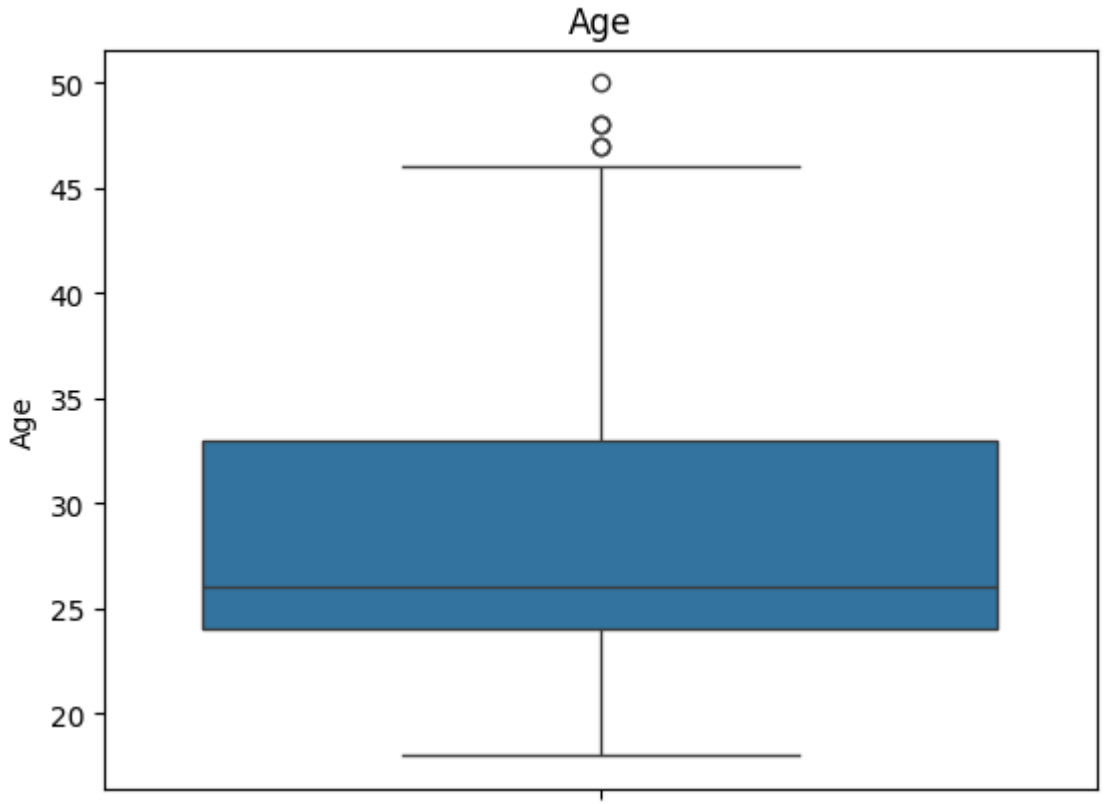

In [254…
```
aerofit.head()
```

Out[254]:

| | Product | Age | Gender | Education | MaritalStatus | Usage | Fitness | Income | Miles | fitness_cat | income_cat | age_cat |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | KP281 | 18 | Male | 14 | Single | 3 | 4 | 29562 | 112 | Good | Low | Young |
| **1** | KP281 | 19 | Male | 15 | Single | 2 | 3 | 31836 | 75 | Good | Low | Young |
| **2** | KP281 | 19 | Female | 14 | Partnered | 4 | 3 | 30699 | 66 | Good | Low | Young |
| **3** | KP281 | 19 | Male | 12 | Single | 3 | 3 | 32973 | 85 | Good | Low | Young |
| **4** | KP281 | 20 | Male | 13 | Partnered | 4 | 2 | 35247 | 47 | Bad | Low | Young |

In [255…
```
sns.boxplot(data = aerofit, y= 'Age')
plt.title("Age")
```

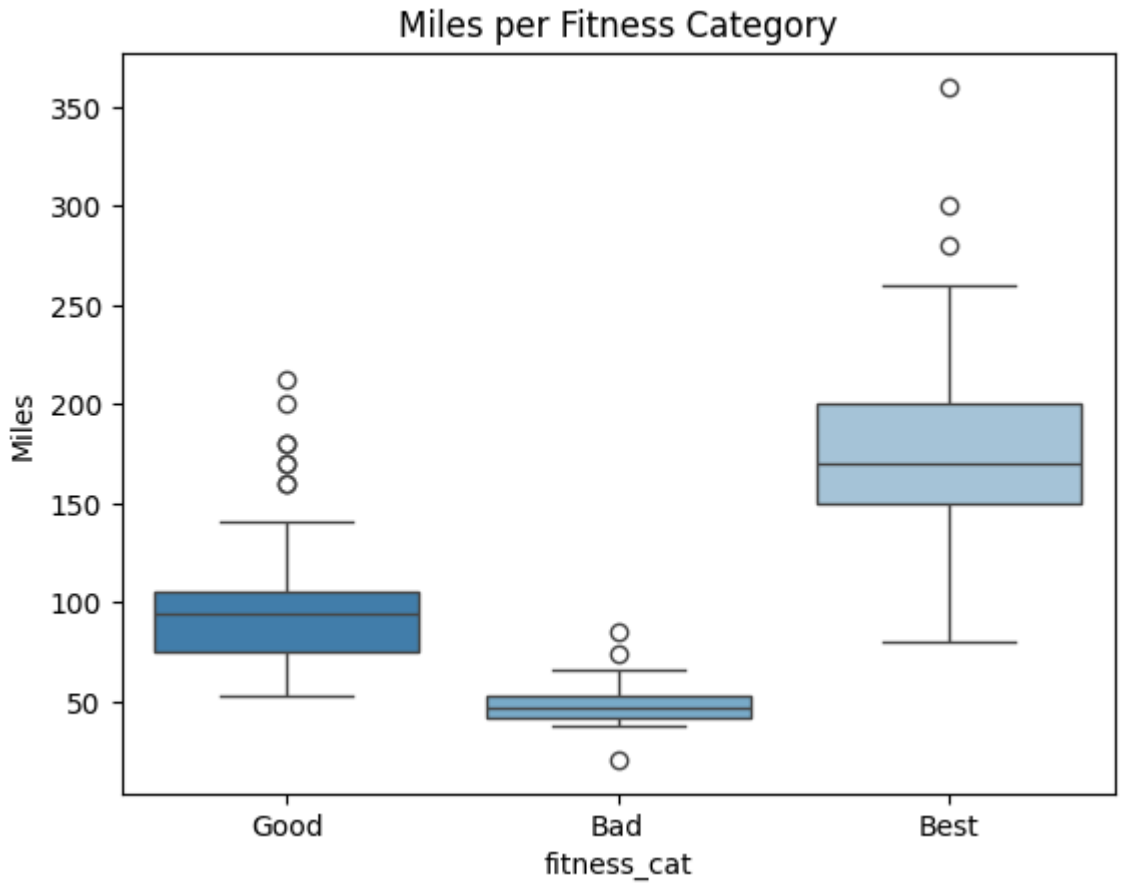
Out[255]: Text(0.5, 1.0, 'Age')




In [256…
```
sns.boxplot(data = aerofit, y= "Miles", x = "fitness_cat", palette = "tab20c")
plt.title("Miles per Fitness Category")
```

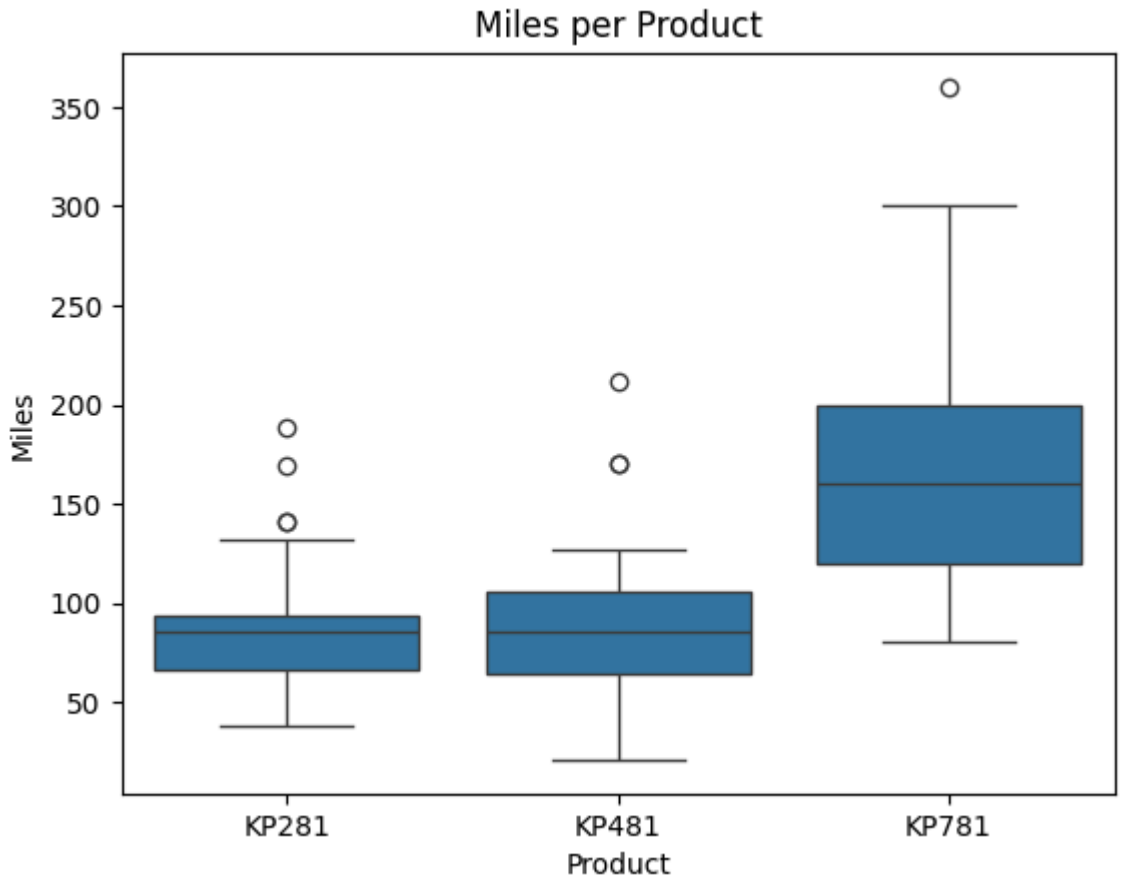
```
<ipython-input-256-c6037c147693>:1: FutureWarning:

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 effect.

  sns.boxplot(data = aerofit, y= "Miles", x = "fitness_cat", palette = "tab20c")
```

Out[256]: Text(0.5, 1.0, 'Miles per Fitness Category')




In [257…
```
sns.boxplot(data = aerofit, x= 'Product', y = 'Miles')
plt.title("Miles per Product")
```

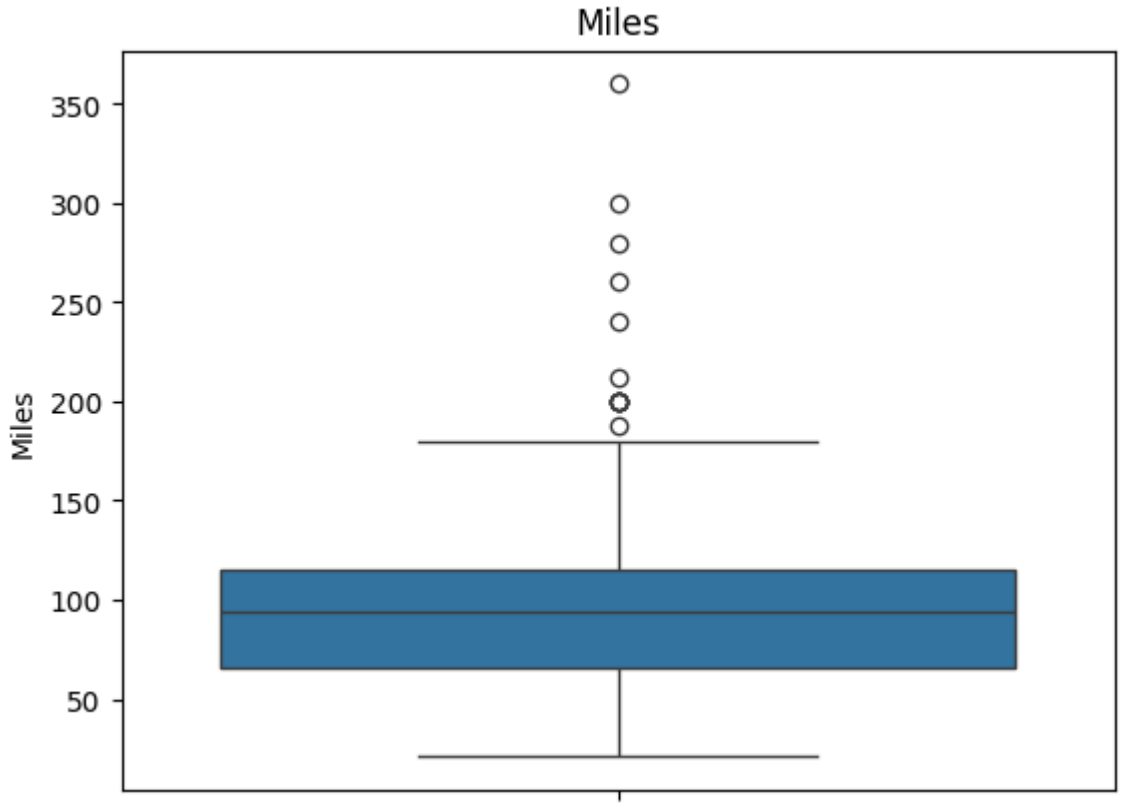
Out[257]:
```
Text(0.5, 1.0, 'Miles per Product')
```




# Checking for Outliers

In [258…
```
sns.boxplot(data = aerofit, y= 'Miles')
plt.title("Miles")
```

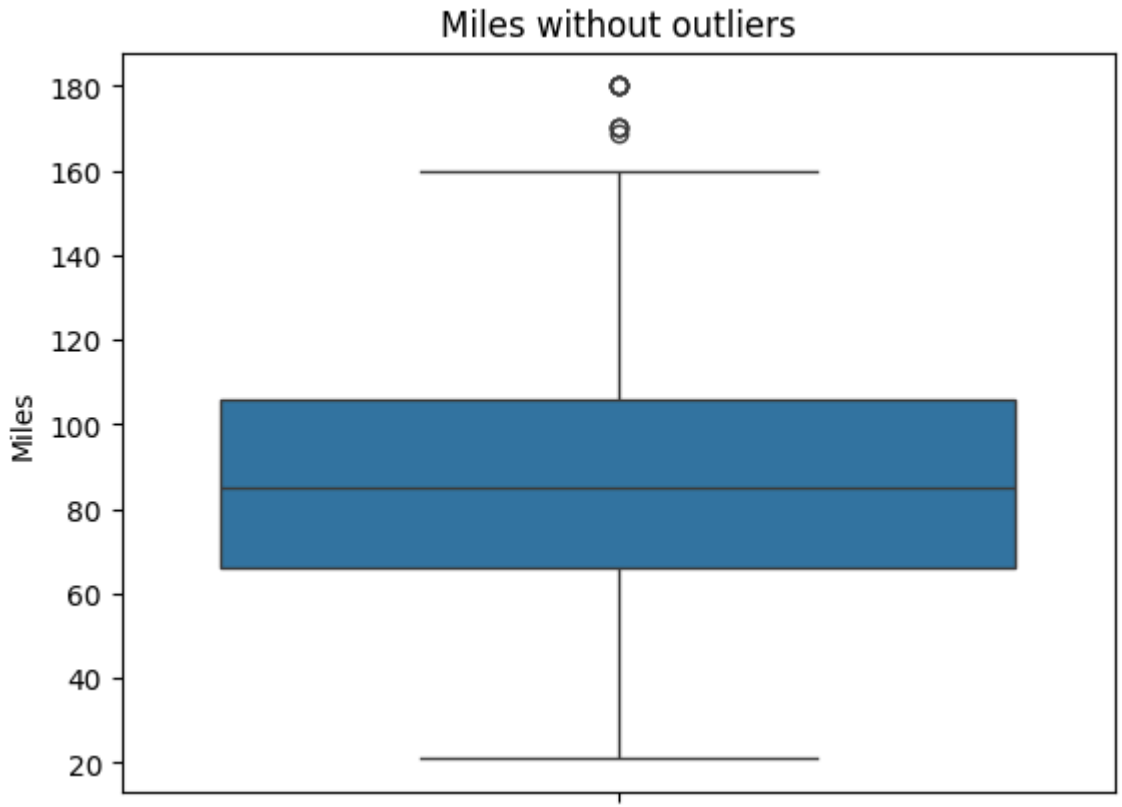
Out[258]:
```
Text(0.5, 1.0, 'Miles')
```




In [259…
```
#Removing outliers
miles_25 = np.percentile(aerofit["Miles"],25)
miles_50 = np.percentile(aerofit["Miles"],50)
miles_75 = np.percentile(aerofit["Miles"],75)
IQR_miles = miles_75- miles_25
upper= miles_75 +IQR_miles*1.5
lower = miles_25- IQR_miles*1.5
miles = aerofit[(aerofit["Miles"]<upper) & (aerofit["Miles"]>lower)]
```

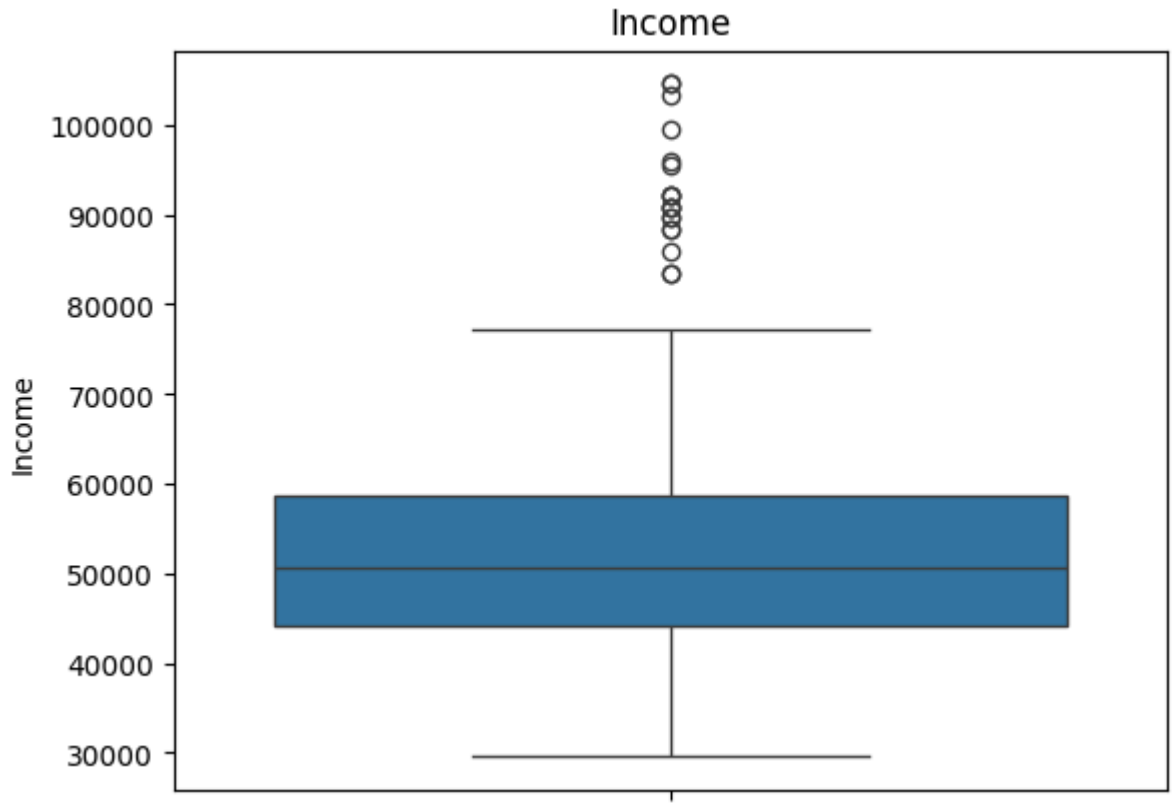
In [260…
```
sns.boxplot(data = miles, y= "Miles")
plt.title("Miles without outliers")
```

Out[260]:
```
Text(0.5, 1.0, 'Miles without outliers')
```




In [261…
```
sns.boxplot(data = aerofit, y= 'Income')
plt.title("Income")
```

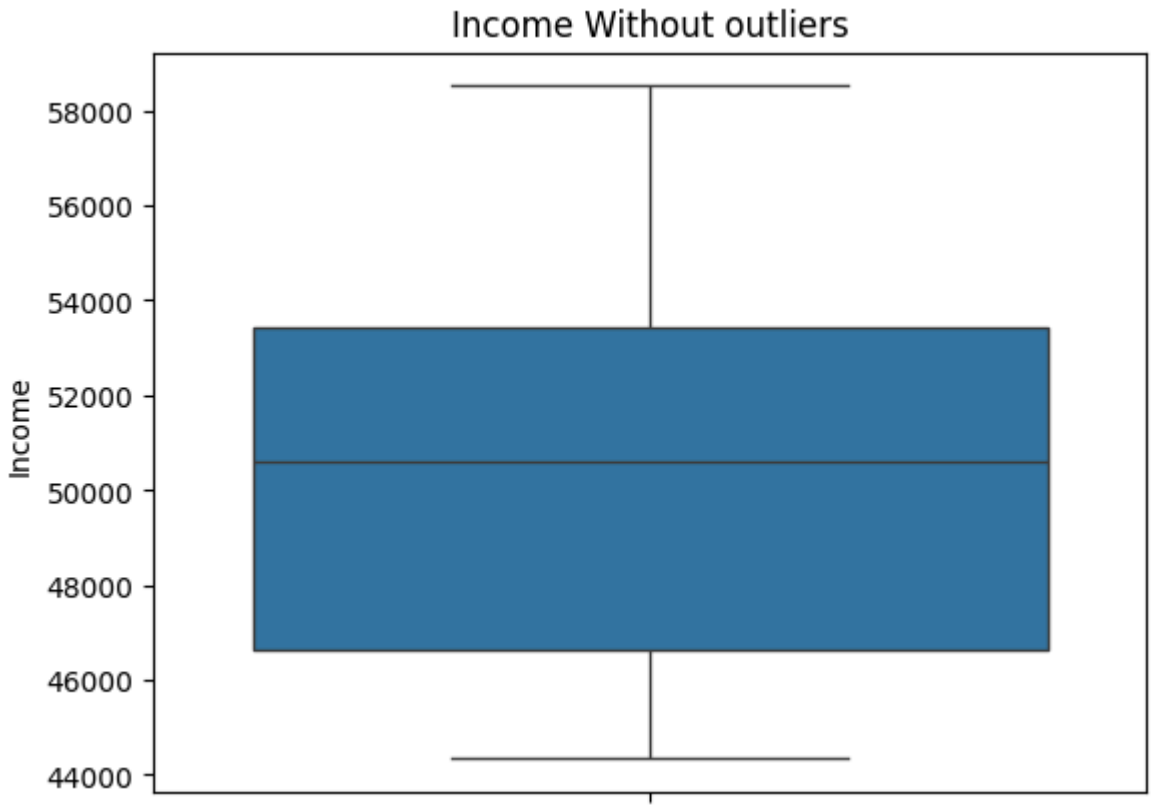
Out[261]:
```
Text(0.5, 1.0, 'Income')
```




In [262…
```
#Removing outliers
income_25 = np.percentile(aerofit["Income"],25)
income_50 = np.percentile(aerofit["Income"],50)
income_75 = np.percentile(aerofit["Income"],75)
IQR_income = income_75-income_25
upper_income= income_75 +IQR_miles*1.5
lower_income = income_25- IQR_miles*1.5
income = aerofit[(aerofit["Income"]<upper_income) & (aerofit["Income"]>lower_income)]
```

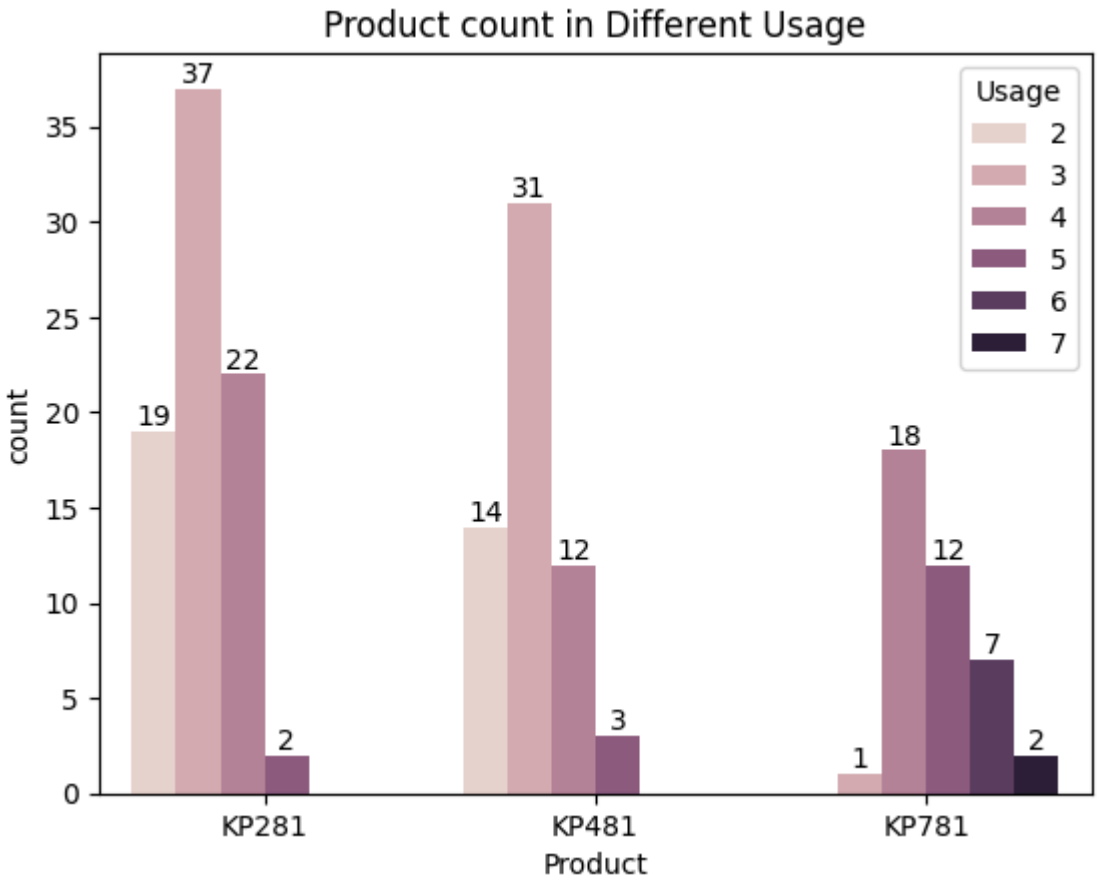
In [263…
```
sns.boxplot(data = income, y= "Income")
plt.title("Income Without outliers")
```

Out[263]:
```
Text(0.5, 1.0, 'Income Without outliers')
```




In [264…
```
usage_count = sns.countplot(data = aerofit, x = "Product", hue = "Usage")
for i in usage_count.containers:
    usage_count.bar_label(i)
plt.title("Product count in Different Usage")
```

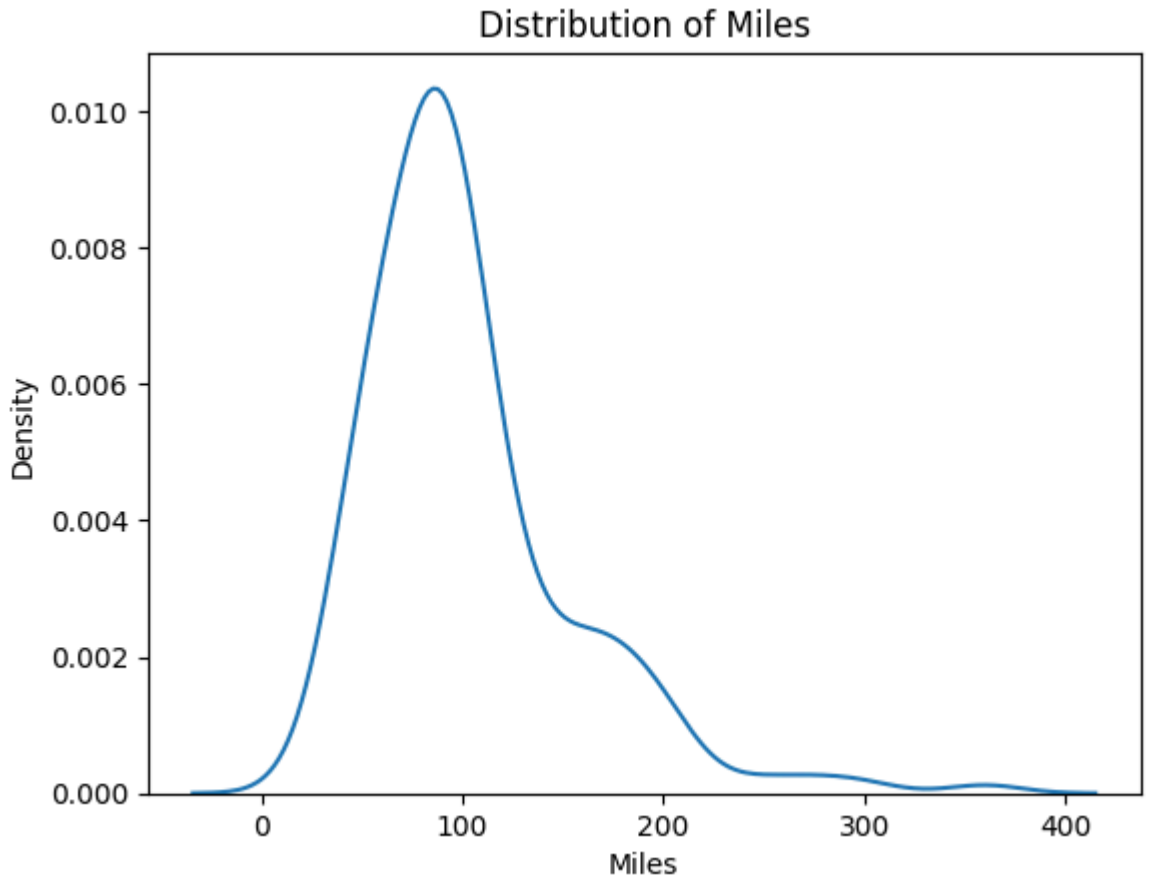
Out[264]:
```
Text(0.5, 1.0, 'Product count in Different Usage')
```




In [265…

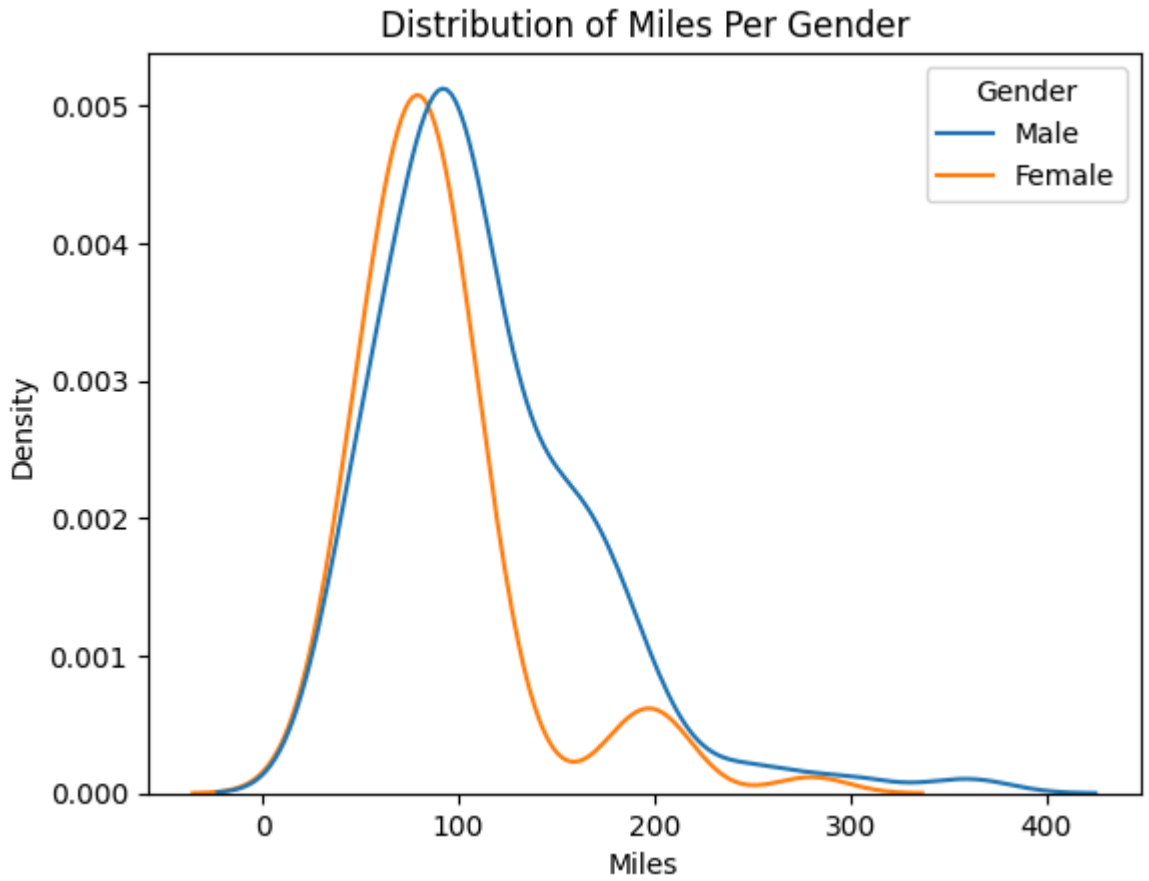
```
sns.kdeplot(aerofit["Miles"],  palette = "tab20")
plt.title("Distribution of Miles")
plt.show()
```

```
<ipython-input-265-5bd18617e03e>:1: UserWarning: Ignoring `palette` because no `hue` variable has been assigned.
  sns.kdeplot(aerofit["Miles"],  palette = "tab20")
```



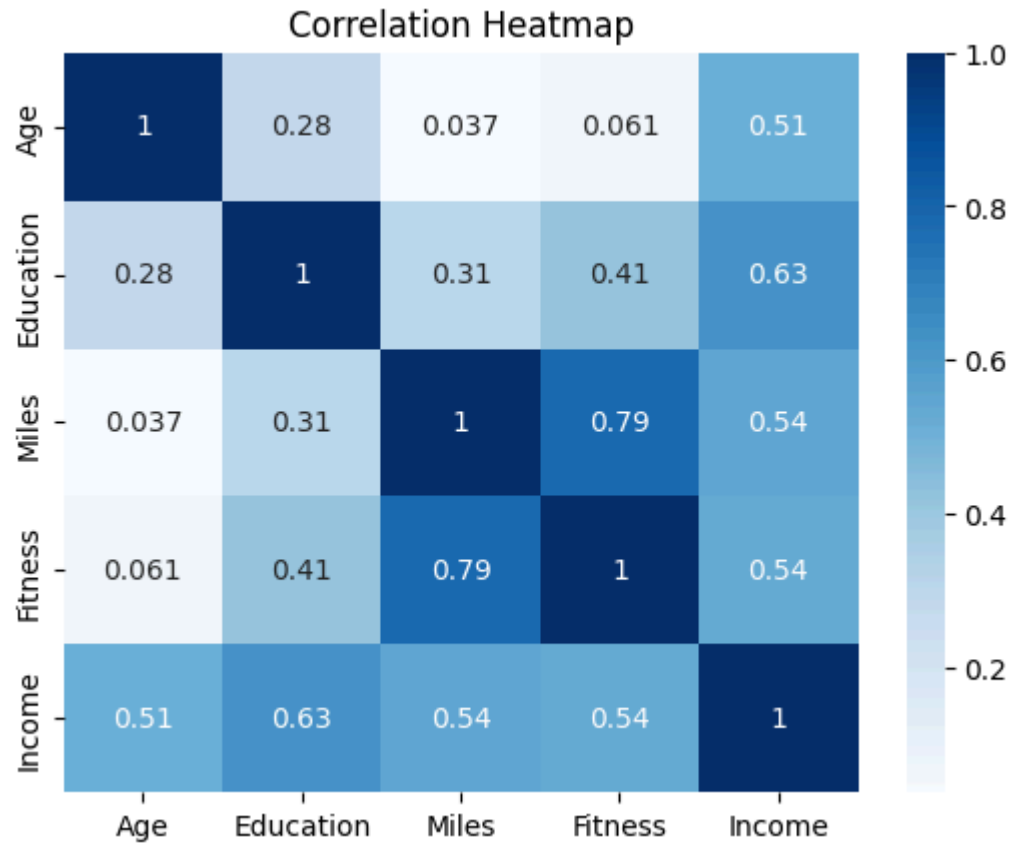

In [266…

```
sns.kdeplot(data = aerofit, x="Miles", hue = "Gender")
plt.title("Distribution of Miles Per Gender")
plt.show()
```




In [267…
```
sns.heatmap(aerofit[['Age','Education','Miles', 'Fitness', "Income"]].corr(), cmap= "Blues", annot =True)
plt.title('Correlation Heatmap')
```

Out[267]:
```
Text(0.5, 1.0, 'Correlation Heatmap')
```

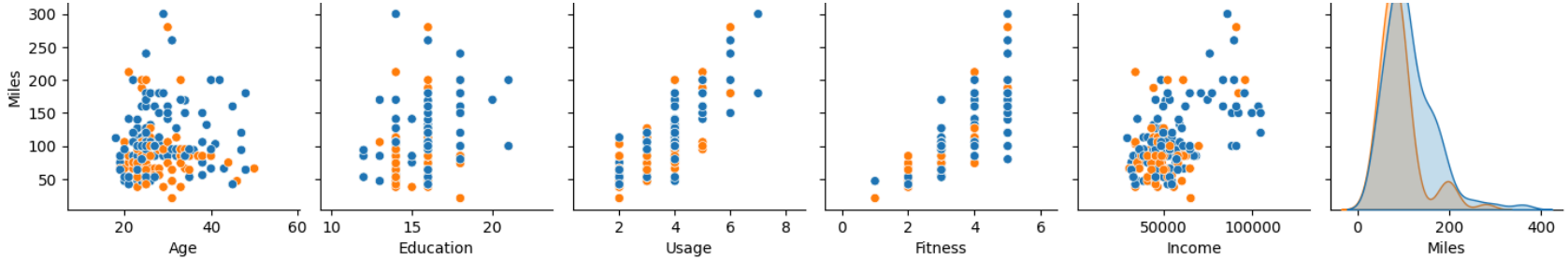



In [268…

```
plt.figure(figsize =(10,5))
sns.pairplot(data = aerofit, hue = "Gender")
plt.show()
```

```
<Figure size 1000x500 with 0 Axes>
```


Age
Education
Usage
Fitness
Income
Gender
Male
Female




# Observation

- Count of KP281 users were found to be more then the other two products.
- Aerofit has a overall customers count of 180 where 104 are males.
- Aerofit has two categories of customers like married/parented and singles. The count of the married customers were found to be more than that of the singes.
- Fitness level 3 has the highest value counts among all the fitness levels.
- Distribution of people was found more between 20 and 30.
- Most of the customers found to be using the threadmills 3 to 4 times a week.
- More customers have 16yrs Education.
- More customers have income between - 40,000 - 60,000
- If the customer expects to walk/run greater than 120 Miles per week, it is more likely that the customer will buy KP781 product.
- The more the customer is fit (fitness >= 3), higher the chances of the customer to purchase the KP781 product.

# Statistical Analysis

# Marginal Probability

In [ ]:
```
product_crosstab = pd.crosstab(index = aerofit["Product"], columns= aerofit["Gender"],margins = True)
product_crosstab
```

In [ ]:
```
males = aerofit[aerofit["Gender"]== "Male"]
females = aerofit[aerofit["Gender"]== "Female"]
p_males = len(males)/len(aerofit)
p_females = len(females)/len(aerofit)
p_females
```

In [ ]:
```
p_males
```

In [ ]:
```
crosstab_result = pd.crosstab(index= aerofit["Product"], columns= aerofit["Usage"], margins  = True)
crosstab_result
```

In [269…
```
usage_3 = aerofit[aerofit["Usage"]==3]
usage_4 = aerofit[aerofit["Usage"]==4]
usage_5 = aerofit[aerofit["Usage"]==5]
```

In [270…
```
p_usage_3= len(usage_3)/len(aerofit)
p_usage_4= len(usage_4)/len(aerofit)
p_usage_5= len(usage_5)/len(aerofit)
print(p_usage_3, p_usage_4, p_usage_5)
```

```
0.38333333333333336 0.28888888888888886 0.09444444444444444
```

In [271…
```
fitness_crosstab = pd.crosstab(index= aerofit["Product"], columns= aerofit["Fitness"], margins = True)
fitness_crosstab
```

Out[271]:

| 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 |

In [272…
```
fitness_3 = aerofit[aerofit["Fitness"]==3]
fitness_4 = aerofit[aerofit["Fitness"]==4]
fitness_5 = aerofit[aerofit["Fitness"]==5]
```

In [273…
```
p_fitness_3= len(fitness_3)/len(aerofit)
p_fitness_4= len(fitness_4)/len(aerofit)
p_fitness_5= len(fitness_5)/len(aerofit)
print(p_fitness_3, p_fitness_4, p_fitness_5)
```

```
0.5388888888888889 0.13333333333333333 0.17222222222222222
```

In [274…
```
pd.crosstab(index = aerofit["Product"],columns = aerofit["MaritalStatus"],margins = True)
```

Out[274]:

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

In [275…
```
parents = aerofit[aerofit["MaritalStatus"]== "Partnered"]
single = aerofit[aerofit["MaritalStatus"]== "Single"]
p_parents = len(parents)/len(aerofit)
p_singles = len(single)/len(aerofit)
p_parents
```

Out[275]:
```
0.5944444444444444
```

In [276…
```
p_singles
```

Out[276]:
```
0.40555555555555556
```

In [277…
```
fitness_cat_crosstab = pd.crosstab(index= aerofit["Product"], columns= aerofit["fitness_cat"], normalize = True)
round(fitness_cat_crosstab,2)
```

Out[277]:

| fitness_cat | Bad | Best | Good |
|---|---|---|---|
| Product | | | |
| **KP281** | 0.08 | 0.01 | 0.35 |
| **KP481** | 0.07 | 0.00 | 0.26 |
| **KP781** | 0.00 | 0.16 | 0.06 |

# Observations

- Probability of female customers buying products = 0.422
- Probability of Male customers buying products = 0.577
- Probability of partnered customers buying the products = 0.594
- Probability of buying products by customers who are single = 0.405
- Probability of customers using any treadmil 3 times a week = 0.383
- Probability of customers using any treadmil 4 times a week = 0.288
- Probaility of customers using treadmil 5 times a week = 0.094
- probability of customers having fitness level 3 = 0.537
- probability of customers having fitness level 4 = 0.13333
- probability of customers having fitness level 5 = 0.1722

# Conditional Probability

# Probability of customers buying treadmill, given fitness level 3

In [278…

```
KP281 = fitness_3[fitness_3["Product"]== "KP281"]
p_KP281_in_fitness3= len(KP281)/len(fitness_3)
p_KP281_in_fitness3
```

Out[278]:

```
0.5567010309278351
```

In [279…
```
KP281 = fitness_3[fitness_3["Product"]== "KP481"]
p_KP281_in_fitness3= len(KP281)/len(fitness_3)
p_KP281_in_fitness3
```

Out[279]:
```
0.4020618556701031
```

In [280…
```
KP281 = fitness_3[fitness_3["Product"]== "KP781"]
p_KP281_in_fitness3= len(KP281)/len(fitness_3)
p_KP281_in_fitness3
```

Out[280]:
```
0.041237113402061855
```

# Probability of buying any threadmil given male customers

In [281…
```
KP281_males = males[males["Product"]== "KP281"]
p_KP281_males = len(KP281_males)/len(males)
p_KP281_males
```

Out[281]:
```
0.38461538461538464
```

In [282…
```
KP481_males = males[males["Product"]== "KP481"]
p_KP481_males = len(KP481_males)/len(males)
p_KP481_males
```

Out[282]:
```
0.2980769230769231
```

In [283…
```
KP781_males = males[males["Product"]== "KP781"]
p_KP781_males = len(KP781_males)/len(males)
p_KP781_males
```

Out[283]:
```
0.3173076923076923
```

# Probability of buying any threadmil given female customers

In [284…
```
KP281_females = females[females["Product"]== "KP281"]
p_KP281_females = len(KP281_females)/len(females)
```

```
p_KP281_females
```

Out[284]: 0.5263157894736842

In [285…
```
KP481_females = females[females["Product"]== "KP481"]
p_KP481_females = len(KP481_females)/len(females)
p_KP481_females
```

Out[285]: 0.3815789473684211

In [286…
```
KP781_females = females[females["Product"]== "KP781"]
p_KP781_females = len(KP781_females)/len(females)
p_KP781_females
```

Out[286]: 0.09210526315789473

# Probability of buying any threadmil given Marital status

In [287…
```
KP281_parents = parents[parents["Product"]=="KP281"]
p_KP281_parents= len(KP281_parents)/len(parents)
p_KP281_parents
```

Out[287]: 0.4485981308411215

In [288…
```
KP481_parents = parents[parents["Product"]=="KP481"]
p_KP481_parents= len(KP481_parents)/len(parents)
p_KP481_parents
```

Out[288]: 0.3364485981308411

In [289…
```
KP781_parents = parents[parents["Product"]=="KP781"]
p_KP781_parents= len(KP781_parents)/len(parents)
p_KP781_parents
```

Out[289]: 0.21495327102803738

In [290…
```
KP281_single = single[single["Product"]=="KP281"]
p_KP281_single= len(KP281_single)/len(single)
p_KP281_single
```

Out[290]:
```
0.4383561643835616
```

In [291…
```
KP481_single = single[single["Product"]=="KP481"]
p_KP481_single= len(KP481_single)/len(single)
p_KP481_single
```

Out[291]:
```
0.3287671232876712
```

In [292…
```
KP781_single = single[single["Product"]=="KP781"]
p_KP781_single= len(KP781_single)/len(single)
p_KP781_single
```

Out[292]:
```
0.2328767123287671
```

# Observation

# Probability of customers buying treadmill, given fitness level 3

- p[kp281|fitness3] = 0.556
- p[kp481|fitness3] = 0.402
- p[kp781|fitness3] = 0.041 #Probability of buying any threadmil given male customers
- p[kp281|male] = 0.384
- p[kp481|male] = 0.298
- p[kp781|male] = 0.317 #Probability of buying any threadmil given female customers
- p[kp281|female] = 0.526
- p[kp481|female] = 0.381
- p[kp781|female] = 0.092 #Probability of buying any threadmil given Marital status
- p[kp281|married] = 0.448
- p[kp481|married] = 0.336
- p[kp781|married] = 0.214
- p[kp281|single] = 0.438
- p[kp481|single] = 0.328

- p[kp781|single] = 0.232

# Customer Profiling

In [293…
```
aerofit["fitness_cat"].value_counts()
```

Out[293]:
```
fitness_cat
Good    121
Best     31
Bad      28
Name: count, dtype: int64
```

In [294…
```
aerofit["income_cat"].value_counts()
```

Out[294]:
```
income_cat
Medium    129
Low        32
High       19
Name: count, dtype: int64
```

In [295…
```
aerofit["age_cat"].value_counts()
```

Out[295]:
```
age_cat
Adult    89
Young    79
Old      12
Name: count, dtype: int64
```

# Insights

1. Out of the 3 products, KP281 is the most used among the customers with highest count and then followed by KP481. We can see that the usage of the high model machine is used very less.
2. Out of all the customers, 57.8% of the customers were found to be males and 42.2% are found to be females.
3. 59.4% are married or parents and 40.6% are singles. Marital Status does not appear to affect product choice, though when looking at KP781, thosewho are partnered have higher fitness levels than those who are single.
4. People with best fitness level were found to be running higher miles and people who are running higher miles are using KP781 machine. So, KP781 users are highly fit and are also high earning customers. This proves that the customers with high fitness and salary have a positive

correlation.

5. Fitness of males is is high compared to females based on the distribution of miles they ran. Most of the customers with good fitness ran more number of miles using the top model KP781.
6. Most of the people are using KP281 and KP481 thrice a week and KP781 machine 4 times a week.
7. KP481 stands between KP781 and KP281 products. People using this product dound have fitness level 3 just like the KP281 customers.

# Recommendations

1. Since most of the customers are using KP281. They have to be maintained in the stock all the time.
2. We can focus on the KP481 customers to buy the KP781 product by giving special offers on the product.
3. Most of the customers are married/parented. Aerofit can conduct a suruvey to identify the reason for singles not buying the product.
4. The customers using the KP281 or KP481 were found to be having less income. So, we can identify the customers with good fitness levels ans low income. Offering them special discounts or coupouns could possible gain the customers for KP781.
5. Most of the customers are from the age group 20- 30. Aerofit can start a campaign on the addressing issues like cardiac diseases and the imporatnce of cardiac health for people above age 30. This would probably create awarness in the people about health and have chances of buying the product.
6. People who are having good fitness levels and running more miles should be encouraged to go for KP781 as it is suitable for running more miles.
7. People using more than 3 times should also be encouraged to buy the KP781 product.