Cardio Good Fitness

Objective:

Explore the dataset and practice extracting basic observations about the data.

1. Come up with a customer profile (characteristics of a customer) of the different products
2. Perform uni-variate and multi-variate analyses
3. Generate a set of insights and recommendations that will help the company in targeting new customers

Key Questions:

1. What is the differences between customers of each product.

Data:

The data is for customers of the treadmill product(s) of a retail store called Cardio Good Fitness. It contains the following variables:

1. Product - the model no. of the treadmill
2. Age - in no of years, of the customer
3. Gender - of the customer
4. Education - in no. of years, of the customer
5. Marital Status - of the customer
6. Usage - Avg. # times the customer wants to use the treadmill every week
7. Fitness - Self rated fitness score of the customer (5 - very fit, 1 - very unfit)
8. Income - of the customer
9. Miles- expected to run

Importing the dataset

In [1]:
```
import warnings
warnings.filterwarnings('ignore') #never print matching warnings
```

In [2]:
```
# Importing the necessary libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [3]:
```
# Reading the dataset
data = pd.read_csv('CardioGoodFitness.csv')
```

In [4]:
```
# copying data to another varaible to avoid any changes to original data
cardio = data.copy()
```

Understanding the structure of the dataset

In [5]:
```
# Visualizing 3 first rows of data
cardio.head(3)
```

Out[5]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	TM195	18	Male	14	Single	3	4	29562	112
1	TM195	19	Male	15	Single	2	3	31836	75
2	TM195	19	Female	14	Partnered	4	3	30699	66

In [6]:
```
# Visualizing 3 last rows of data
cardio.tail(3)
```

Out[6]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
177	TM798	45	Male	16	Single	5	5	90886	160
178	TM798	47	Male	18	Partnered	4	5	104581	120
179	TM798	48	Male	18	Partnered	4	5	95508	180

Observations

- Product contains the model number of the treadmill
- Usage contain the avarege number of times the customer wants to use the treadmill every week
- Fitness its a self rated fitness score of the customer (5 - very fit, 1 - very unfit)
- Miles contain the expected miles to run per week
- The variables: Product, Gender and, MaritalStatus are categorical variable

In [7]:
```
# shape of the dataset
cardio.shape
```

Out[7]: (180, 9)

- The dataset has 180 rows and 9 columns.

In [8]:
```
# dataframe Cardio informations
cardio.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
```

Observations

- All column have 180 observations non-null which indicat that there are no missing values in it *(treatment of missing values is not necessary)*.
- Product, Gender, and MaritalStatus should be categorical varaibles .

In [9]:
```
# Double checking missing values, once using info were possible to see that t
cardio.isna().sum()
```

Out[9]:
```
Product          0
Age              0
Gender           0
Education        0
MaritalStatus    0
Usage            0
Fitness          0
Income           0
Miles            0
dtype: int64
```

Data Preprocessing

In [10]:
```
# converting categorical columns to categorical type
cardio['Product'] = cardio.Product.astype('category')
cardio['Gender'] = cardio.Gender.astype('category')
cardio['MaritalStatus'] = cardio.MaritalStatus.astype('category')
```

In [11]:
```
cardio.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    int64
 4   MaritalStatus  180 non-null    category
 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: category(3), int64(6)
memory usage: 9.4 KB
```

In [12]:
```
# checking descriptive statistics
# include all means even the ones that is not numerical like categorical
cardio.describe(include='all')
```

Out[12]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	
count	180	180.000000	180	180.000000	180	180.000000	180.000000	
unique	3	NaN	2	NaN	2	NaN	NaN	
top	TM195	NaN	Male	NaN	Partnered	NaN	NaN	
freq	80	NaN	104	NaN	107	NaN	NaN	
mean	NaN	28.788889	NaN	15.572222	NaN	3.455556	3.311111	53
std	NaN	6.943498	NaN	1.617055	NaN	1.084797	0.958869	16
min	NaN	18.000000	NaN	12.000000	NaN	2.000000	1.000000	29
25%	NaN	24.000000	NaN	14.000000	NaN	3.000000	3.000000	44
50%	NaN	26.000000	NaN	16.000000	NaN	3.000000	3.000000	50

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	
75%	NaN	33.000000	NaN	16.000000	NaN	4.000000	4.000000	58(
max	NaN	50.000000	NaN	21.000000	NaN	7.000000	5.000000	104

Cardio Good Fitness *Customer Profile*

In [13]:
```
#Creating a overview customer profile, analasyng Average, Median, Min and Max

cardio_profile = cardio.describe().T #function to describe numerical variable

cardio_profile.drop(labels=['count','std','25%','75%'], axis=1, inplace=True)

columns_rename={'mean':'Average','min':'Min', '50%':'Median','max':'Max'} #re
cardio_profile.rename(columns = columns_rename, inplace=True)

cardio_profile #showing profile
```

Out[13]:

	Average	Min	Median	Max
Age	28.788889	18.0	26.0	50.0
Education	15.572222	12.0	16.0	21.0
Usage	3.455556	2.0	3.0	7.0
Fitness	3.311111	1.0	3.0	5.0
Income	53719.577778	29562.0	50596.5	104581.0
Miles	103.194444	21.0	94.0	360.0

In [14]:
```
# analyzing customers by product
product_counts = cardio['Product'].value_counts()
product_counts
```

Out[14]:
```
TM195    80
TM498    60
TM798    40
Name: Product, dtype: int64
```

In [15]:
```
# analyzing customers by gender
gender_counts = cardio['Gender'].value_counts()
gender_counts
```

Out[15]:
```
Male      104
Female     76
Name: Gender, dtype: int64
```

In [16]:
```
# analyzing customers by Marital Status
marital_counts = cardio['MaritalStatus'].value_counts()
marital_counts
```

Out[16]:
```
Partnered    107
Single        73
Name: MaritalStatus, dtype: int64
```

TM195 *Customer Profile*

In [17]:
```
#Creating a product profile, analasyng Average, Median, Min and Max
```

```
tm195_profile = cardio[cardio['Product']=='TM195'] #filter by product
tm195_profile.head()
```

Out[17]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
0	TM195	18	Male	14	Single	3	4	29562	112
1	TM195	19	Male	15	Single	2	3	31836	75
2	TM195	19	Female	14	Partnered	4	3	30699	66
3	TM195	19	Male	12	Single	3	3	32973	85
4	TM195	20	Male	13	Partnered	4	2	35247	47

In [18]:

```
tm195_profile_des = tm195_profile.describe().T #function to describe numerica

tm195_profile_des.drop(labels=['count','std','25%','75%'], axis=1, inplace=Tr

columns_rename={'mean':'Average','min':'Min', '50%':'Median','max':'Max'} #re
tm195_profile_des.rename(columns = columns_rename, inplace=True)

tm195_profile_des #showing profile
```

Out[18]:

	Average	Min	Median	Max
Age	28.5500	18.0	26.0	50.0
Education	15.0375	12.0	16.0	18.0
Usage	3.0875	2.0	3.0	5.0
Fitness	2.9625	1.0	3.0	5.0
Income	46418.0250	29562.0	46617.0	68220.0
Miles	82.7875	38.0	85.0	188.0

Observations

- Customers with age on range 18 to 50, but mean 28.6;
- Education mean 15 years;
- Usage and Fitness average 3;
- Miles less than 190 per week;
- Seems to have a moderate to low Use expectation.

TM498 *Customer Profile*

In [19]:

```
#Creating a product profile, analasyng Average, Median, Min and Max

tm498_profile = cardio[cardio['Product'] == 'TM498'] #filter by product
tm498_profile.head()
```

Out[19]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
80	TM498	19	Male	14	Single	3	3	31836	64
81	TM498	20	Male	14	Single	2	3	32973	53
82	TM498	20	Female	14	Partnered	3	3	34110	106
83	TM498	20	Male	14	Single	3	3	38658	95
84	TM498	21	Female	14	Partnered	5	4	34110	212

In [20]:
```
tm498_profile_des = tm498_profile.describe().T #function to describe numerica

tm498_profile_des.drop(labels=['count','std','25%','75%'],axis=1,inplace=True

columns_rename={'mean':'Average','min':'Min', '50%':'Median','max':'Max'} #re
tm498_profile_des.rename(columns = columns_rename, inplace=True)

tm498_profile_des #showing profile
```

Out[20]:

	Average	Min	Median	Max
Age	28.900000	19.0	26.0	48.0
Education	15.116667	12.0	16.0	18.0
Usage	3.066667	2.0	3.0	5.0
Fitness	2.900000	1.0	3.0	4.0
Income	48973.650000	31836.0	49459.5	67083.0
Miles	87.933333	21.0	85.0	212.0

Observations

- Customers with age on range 19 to 48, but mean 28.9;
- Education mean 15 years;
- Usage and Fitness average 3;
- Miles less than 215 per week;
- Seems to have a moderate to low use expectation but more miles per week compared to TM195.

TM798 *Customer Profile*

In [21]:
```
tm798_profile = cardio[cardio['Product']== 'TM798']
tm798_profile.head()
```

Out[21]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
140	TM798	22	Male	14	Single	4	3	48658	106
141	TM798	22	Male	16	Single	3	5	54781	120
142	TM798	22	Male	18	Single	4	5	48556	200
143	TM798	23	Male	16	Single	4	5	58516	140
144	TM798	23	Female	18	Single	5	4	53536	100

In [22]:
```
tm798_profile_des = tm798_profile.describe().T
tm798_profile_des.drop(labels=['count','std','25%','75%'],axis=1,inplace=True

columns_rename={'mean':'Average','min':'Min', '50%':'Median','max':'Max'} #re
tm798_profile_des.rename(columns = columns_rename, inplace=True)

tm798_profile_des #showing profile
```

Out[22]:

	Average	Min	Median	Max
Age	29.100	22.0	27.0	48.0
Education	17.325	14.0	18.0	21.0

	Average	Min	Median	Max
Usage	4.775	3.0	5.0	7.0
Fitness	4.625	3.0	5.0	5.0
Income	75441.575	48556.0	76568.5	104581.0
Miles	166.900	80.0	160.0	360.0

Observations

- Customers with age on range 22 to 48, but mean 29;
- Education mean 17 years and high income (They might be correlated);
- High expectation of use, minimun 3 times a week;
- Highest Usage and Fitness average if compare with the others product.

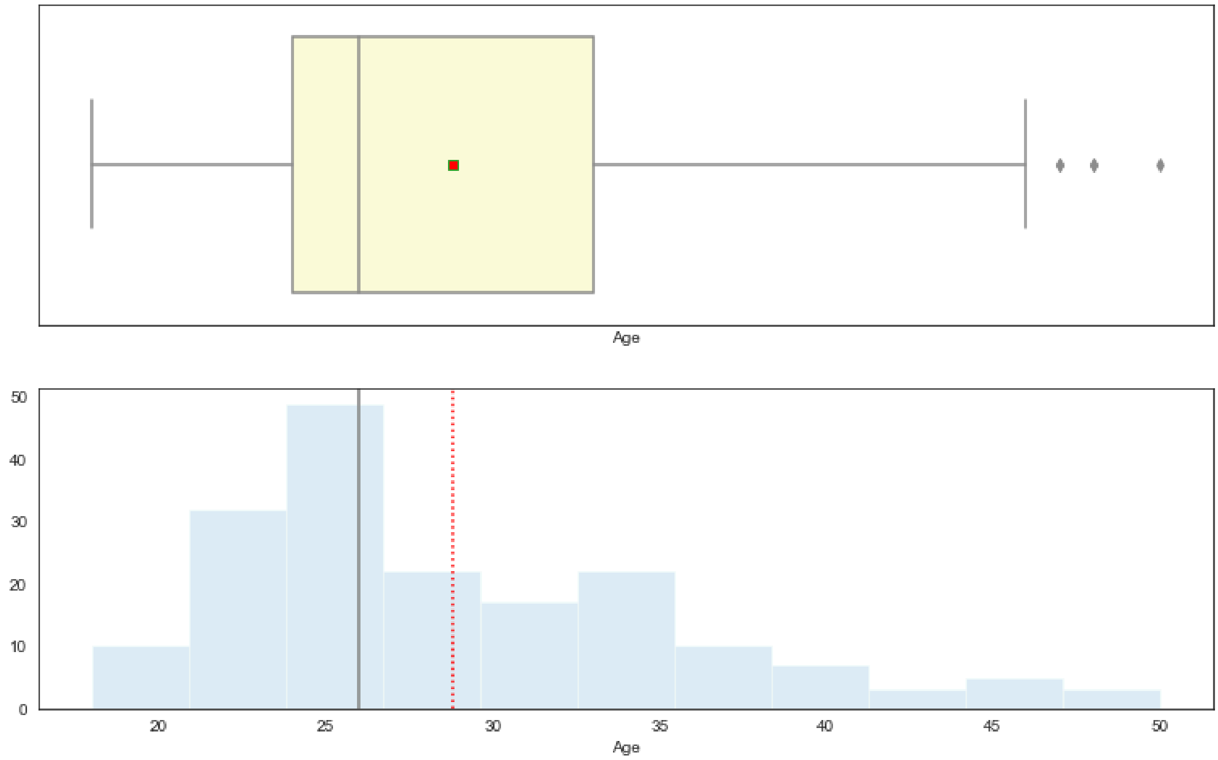
EDA

Uni-variate: Exploring the numerical variables

In [23]:
```
# A function to create boxplot and histogram for any input numerical

def hist_boxplot(dataframe, figsize=(13,8), bins = None):
    """
    This function takes the numerical column as the input and returns the box
    dataframe: 1-d feature array
    figsize: size of fig (default (13,8))
    bins: number of bins (default None / auto)
    """


    # Figure aesthetics
    sns.set_style("white")

    # Creating the 2 subplots
    fig, (ax_box, ax_hist) = plt.subplots(nrows = 2, sharex = True, figsize =

    # Boxplot will be created and a red square will indicate the mean value o
    sns.boxplot(dataframe, ax=ax_box, showmeans=True, meanprops={"marker":"s"

    # For histogram
    sns.distplot(dataframe, kde=F, ax=ax_hist,
                 color='lightblue', bins=bins) if bins else sns.distplot(data
                                                                            kde=

    # Add mean to the histogram
    ax_hist.axvline(np.mean(dataframe), color='r', linestyle='dotted')
    # Add median to the histogram
    ax_hist.axvline(np.median(dataframe), color='gray', linestyle='solid')
```

Observations on Age

In [24]:
```
hist_boxplot(cardio.Age)
```

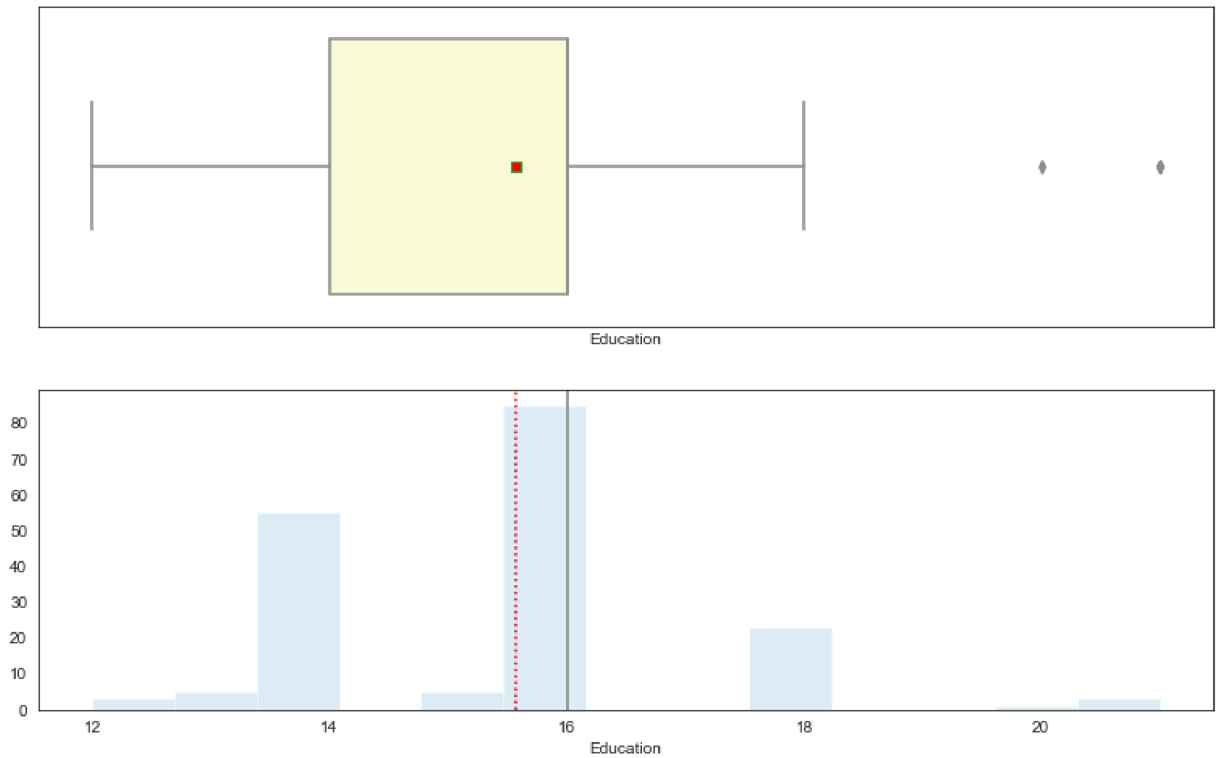


Observations

- The distribution of Age is right skewed, showing that user are concentrate between 18 to 30 years old.
- Median (26) and mean (29) are nearby
- There are some outliers in this variable, customers around 45 to 50 years old.

Observations on Education

In [25]:
```
hist_boxplot(cardio.Education)
```

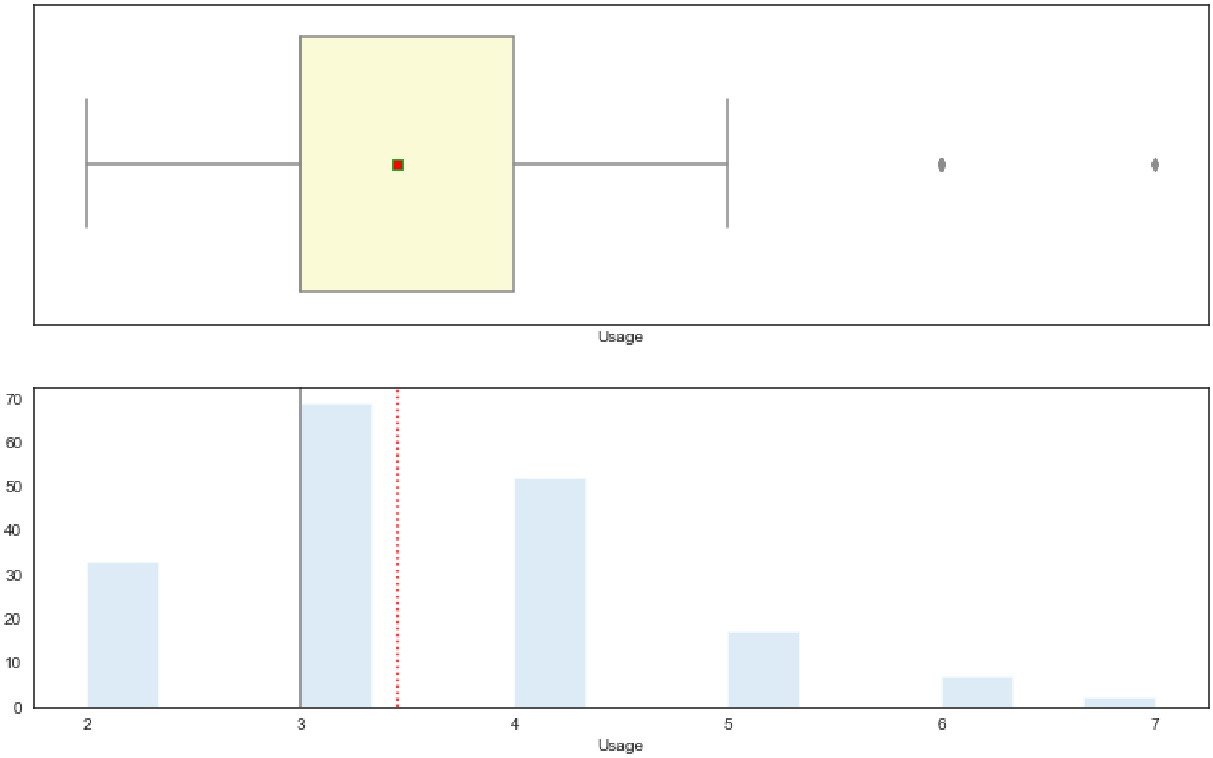


Observations

- Education is concentrated around 16 year.
- Median and mean are almost the same, showing no skew.
- There are some outliers in this variable, showing 20 and 21 years of education.

Observations on Usage

In [26]:
```
hist_boxplot(cardio.Usage)
```

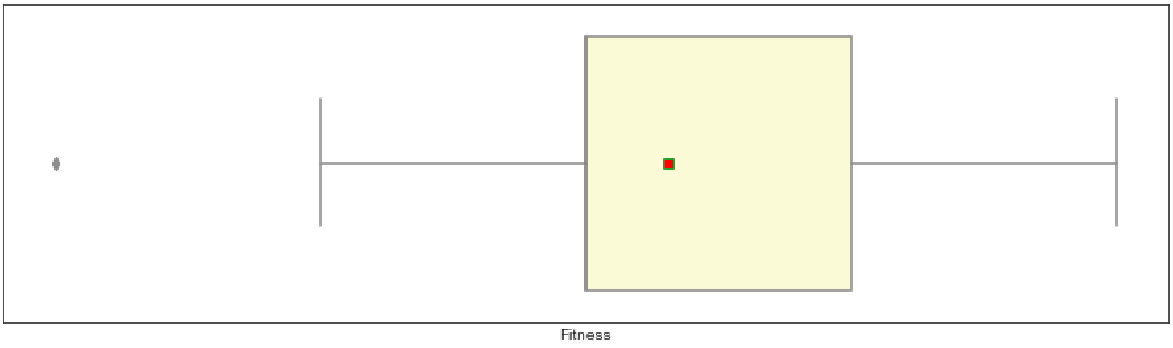


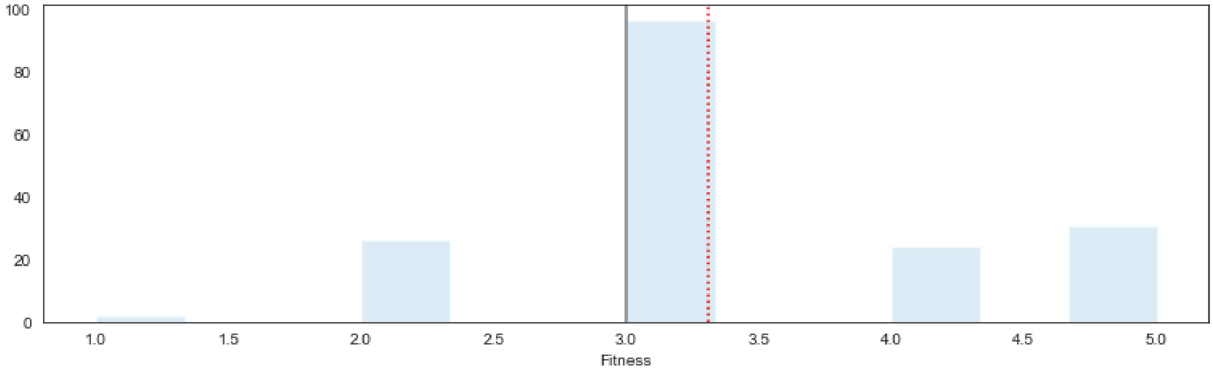
Observations

- Usage it is showing a positive skew, concentrated between 3 and 4 times per week;
- There are some outliers in this variable, showing 6 and 7 times/week using the treadmill.

Observations on Fitness

In [27]:
```
hist_boxplot(cardio.Fitness)
```



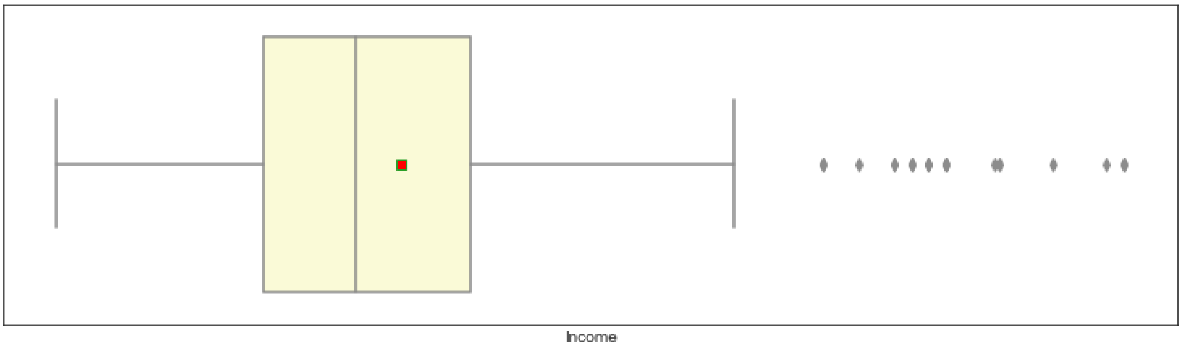


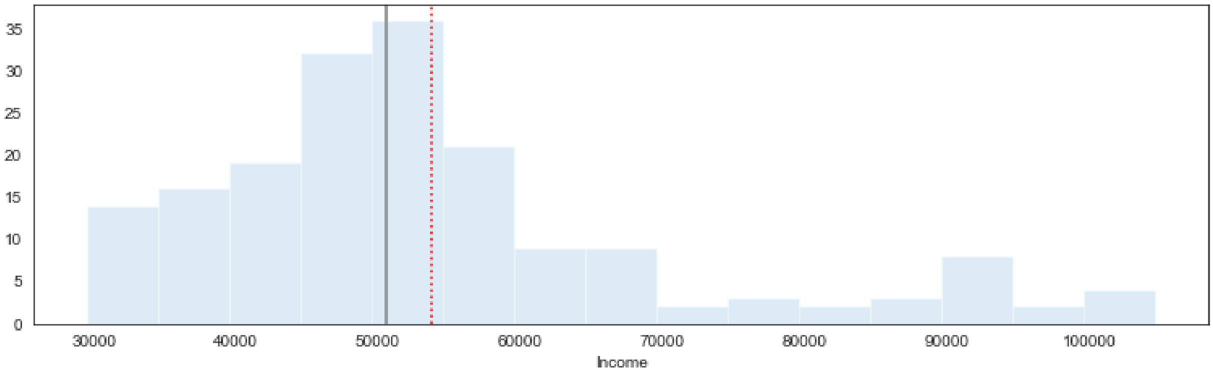
Observations

- Self rate score is concentrated around 3, showing moderate Fitness.
- Median (3) and mean (3.3) are almost the same, showing no skewness.
- There are 2 customers that self rate themself as 1, and they are the outliers.

Observations on Income

In [28]:
```
hist_boxplot(cardio.Income)
```





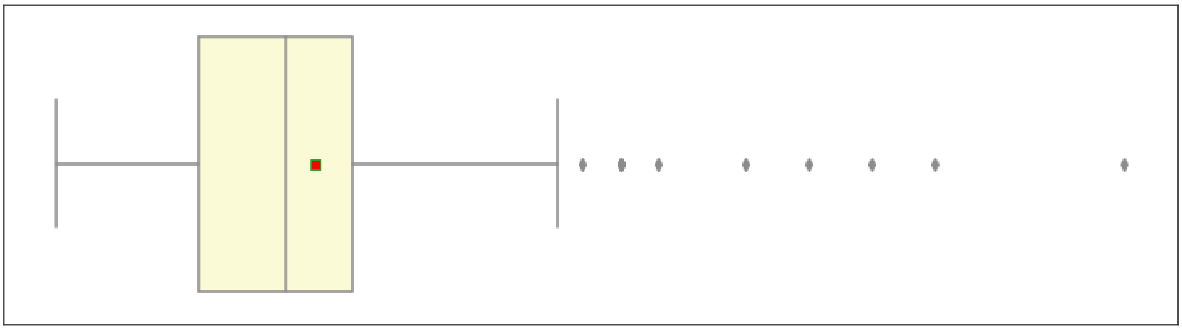
Observations

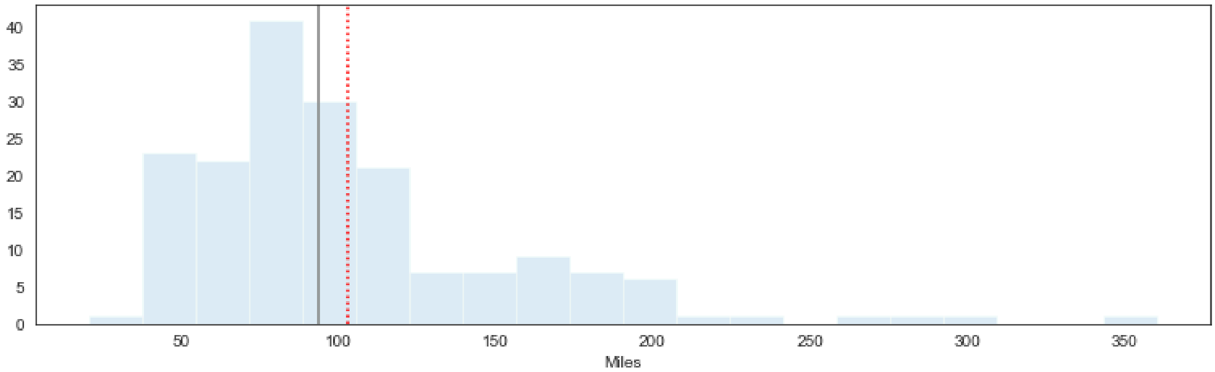
- Income has a right skwness,showing a lot of outliers on the right side;

- The mean income of Customers is around $50k, but the maximum is around 105k.

Observations on Miles

In [29]:
```
hist_boxplot(cardio.Miles)
```





Observations

- There are some outliers on Miles equal to or greater than 200;
- Miles has a right skewness and mean around 100.

In [30]:
```
# Filtering the data to see how many customers expecting to run more than 199
outliers_miles = cardio[cardio['Miles'] > 199]
outliers_miles['Miles'].value_counts().sum()
```

Out[30]: 12

Uni-variate: Exploring the categorical variables

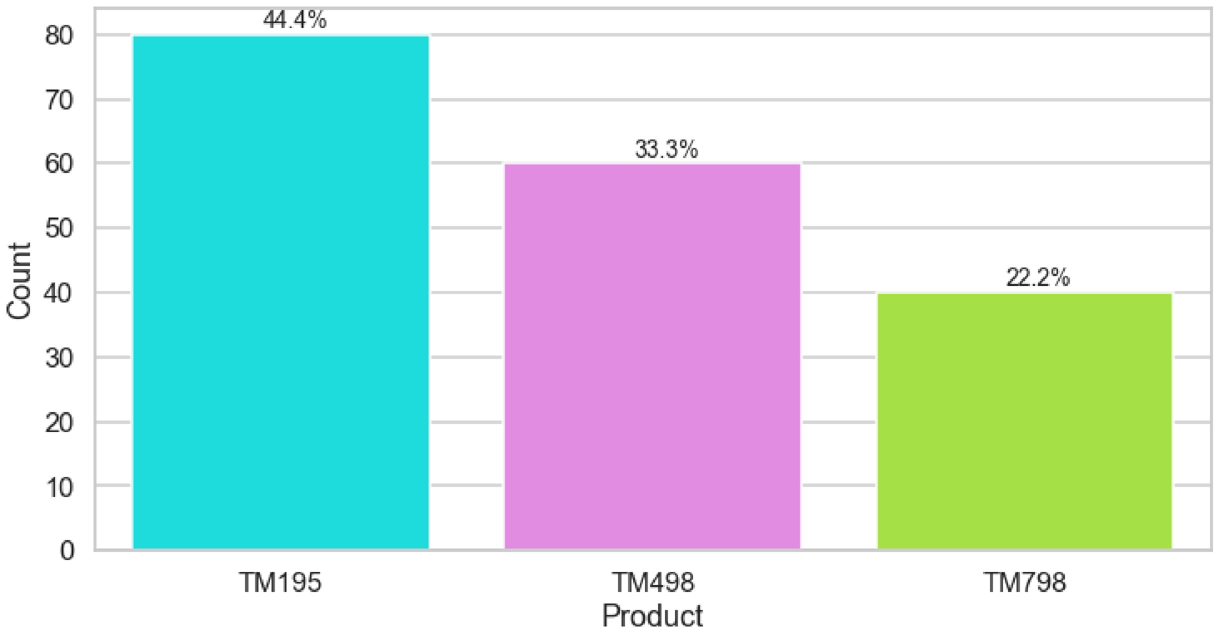
In [31]:
```
# Function to create barplots that indicate percentage for each category.

def bar_perc(dataframe, xlabel_df, colors, ylabel_df = 'Count'):
    """
    This function takes the category column as the input and returns the barp
    dataframe: 1-d categorical feature array
    xlabel_df: x axis label
    ylabel_df: y axis label (default 'Count')
    colors: list of colors to use for the different variables
    """


    # Figure aesthetics
    sns.set_style("whitegrid")
    sns.set_context("talk")
```

```
    # Plot informations
    plt.figure(figsize=(12,6))
    plot_df = sns.countplot(dataframe, palette= colors)
    plt.xlabel(xlabel_df)
    plt.ylabel(ylabel_df)

    # Calculating the length of the column
    total = len(dataframe)

    # Looping to calculate percentage of each class of the category and annot
    for cat in plot_df.patches:
        percentage = '{:.1f}%'.format(100 * cat.get_height()/total)

        #setting plot annotate location and size
        x = cat.get_x() + cat.get_width() / 2 - 0.05
        y = cat.get_y() + cat.get_height() + 1
        plot_df.annotate(percentage, (x, y), size = 14)
```

Observations on Product

In [32]:

```
# List of colors to use for the different products
colors = ['cyan', 'violet', 'greenyellow']

# Using the functio to plot barplot wtih percentage values
bar_perc(cardio['Product'],'Product',colors)
```



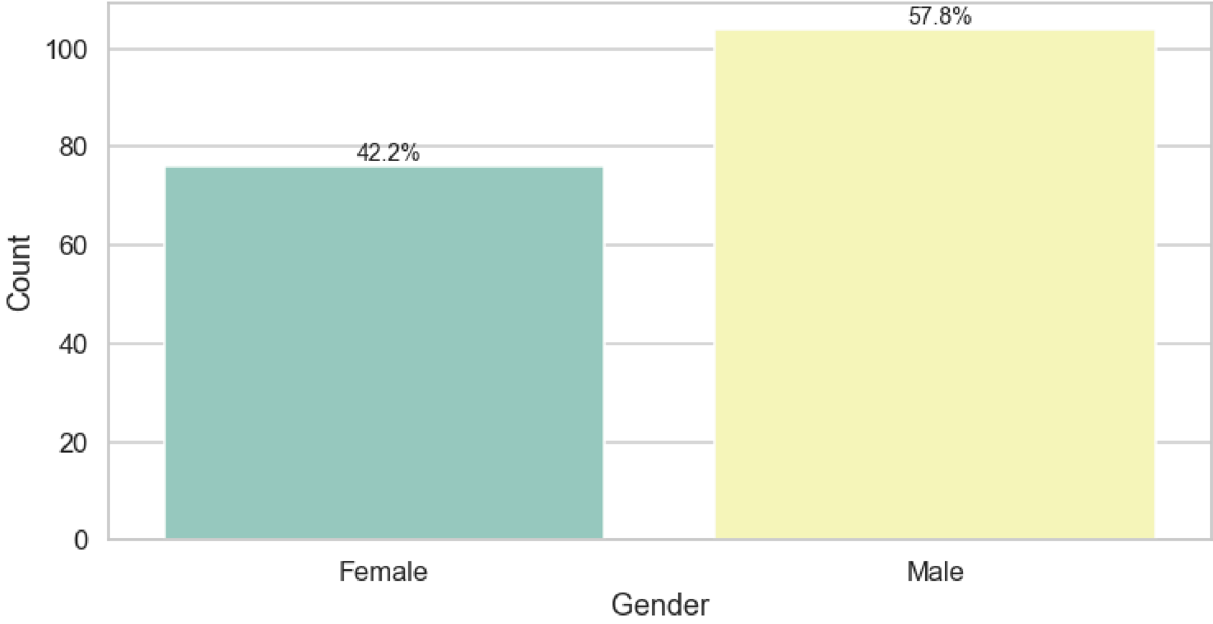
Observations

- The store has 3 models of TreadMill;
- TM195 is the most popular TreadMill, responsible for 44.4% of sales;
- TM498 is the second popular TreadMill and Tm798 the third.

Observations on Gender

In [33]:

```
# List of colors to use for the different gender
colors = "Set3"

# Using the functio to plot barplot wtih percentage values
bar_perc(cardio['Gender'],'Gender',colors)
```

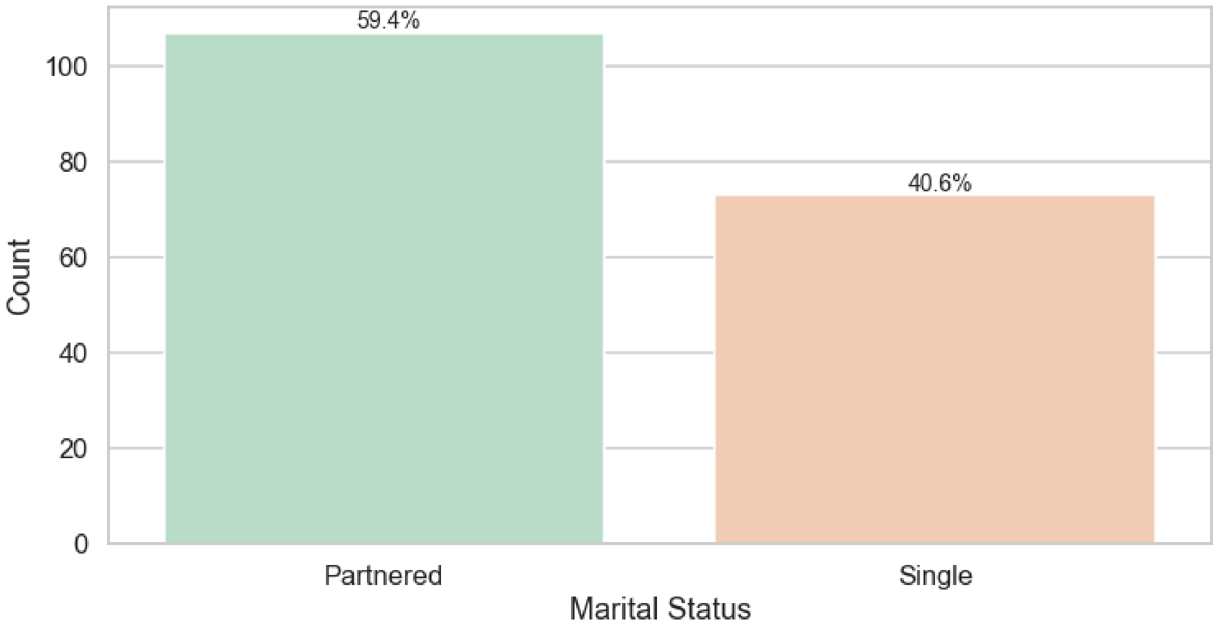


Observations

- 57.8% of customers are Male, showing that Male are buying more than Female.

Observations on Marital Status

In [34]:
```
# List of colors to use for the different Marital Status
colors = "Pastel2"

# Using the functio to plot barplot wtih percentage values
bar_perc(cardio['MaritalStatus'], 'Marital Status',colors)
```



Observations

- 59.4% of customers are Partnered.

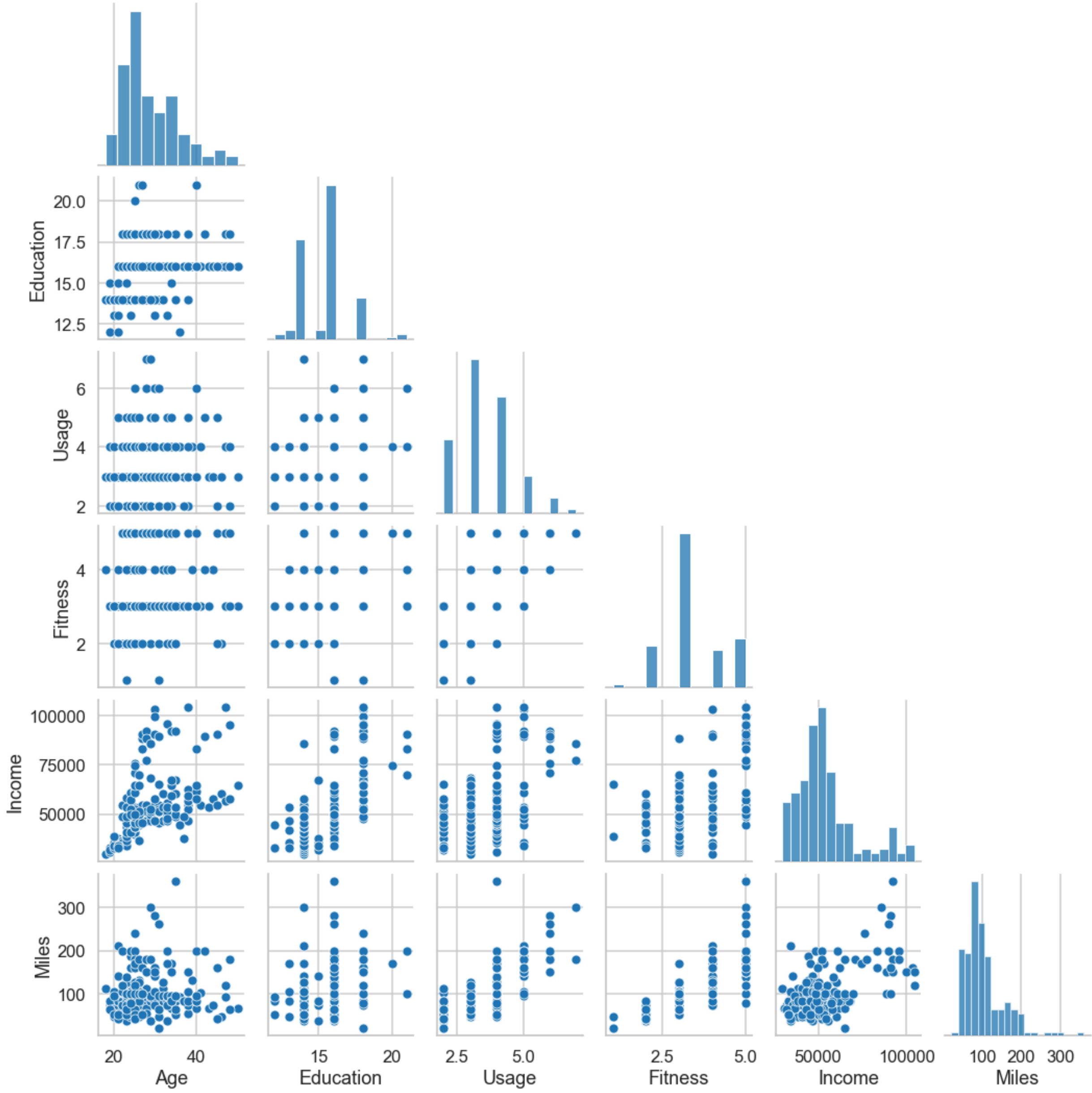
Bivariate / Multivariate: Understanding relationship between variables

In [35]:
```
# Understanding correlation among numerical variables
num_var = ['Age','Education','Usage','Fitness','Income', 'Miles']

corr = cardio[num_var].corr()

# Ploting the heatmap
plt.figure(figsize=(10,8))
sns.heatmap(corr, annot=True,cmap='Greens',fmt=".1f")
```

Out[35]: <AxesSubplot:>

In [36]:
```
# Ploting Bivariate Scatter Plots
sns.pairplot(cardio[num_var], corner=True)
```

Out[36]: <seaborn.axisgrid.PairGrid at 0x1f9178bed60>



Observations

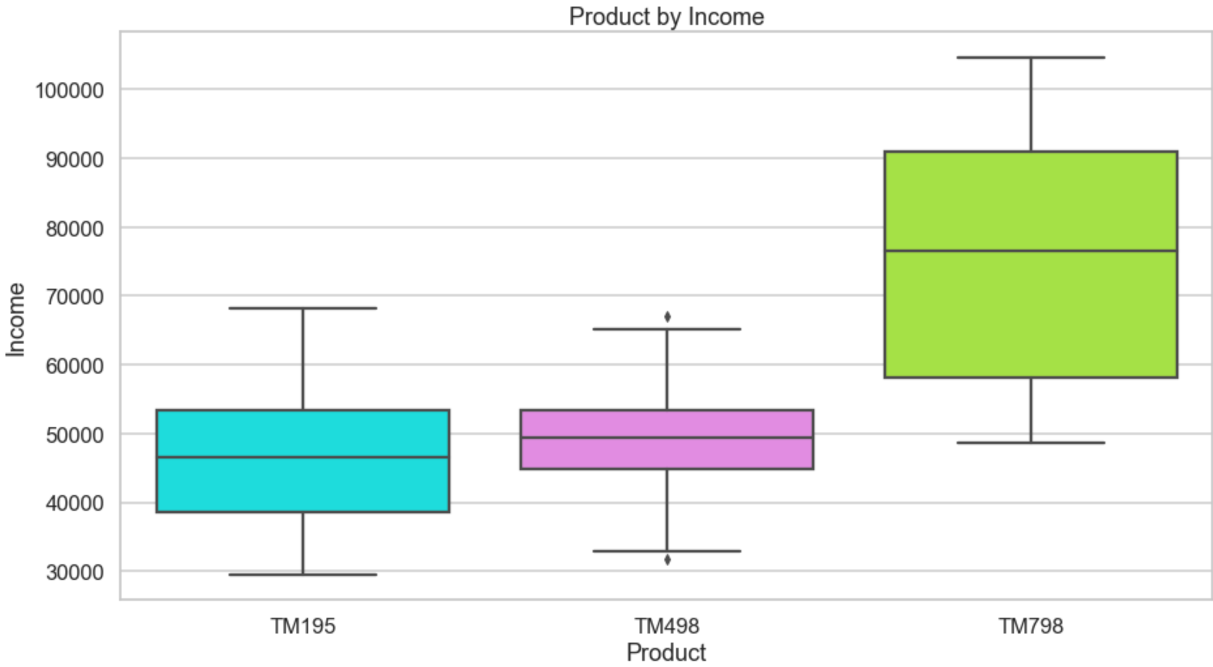
- Age and Education is positive correlated with Income. This is an expected result;
- Fitness has a high positive correlation with Usage and Miles;
- It does not seem to be a strong relationship between Age and Fitness.

Customer Profile

- **Product vs Income**

In [37]:
```
# List of colors to use for the different products
colors = ['cyan', 'violet', 'greenyellow']

# Ploting boxplot for analysis about correlation between Product and Income
plt.figure(figsize=(15,8))

sns.boxplot(cardio['Product'],cardio['Income'], palette = colors)

#Setting Labels
plt.ylabel('Income')
plt.xlabel('Product')
plt.title('Product by Income')
```

Out[37]: Text(0.5, 1.0, 'Product by Income')



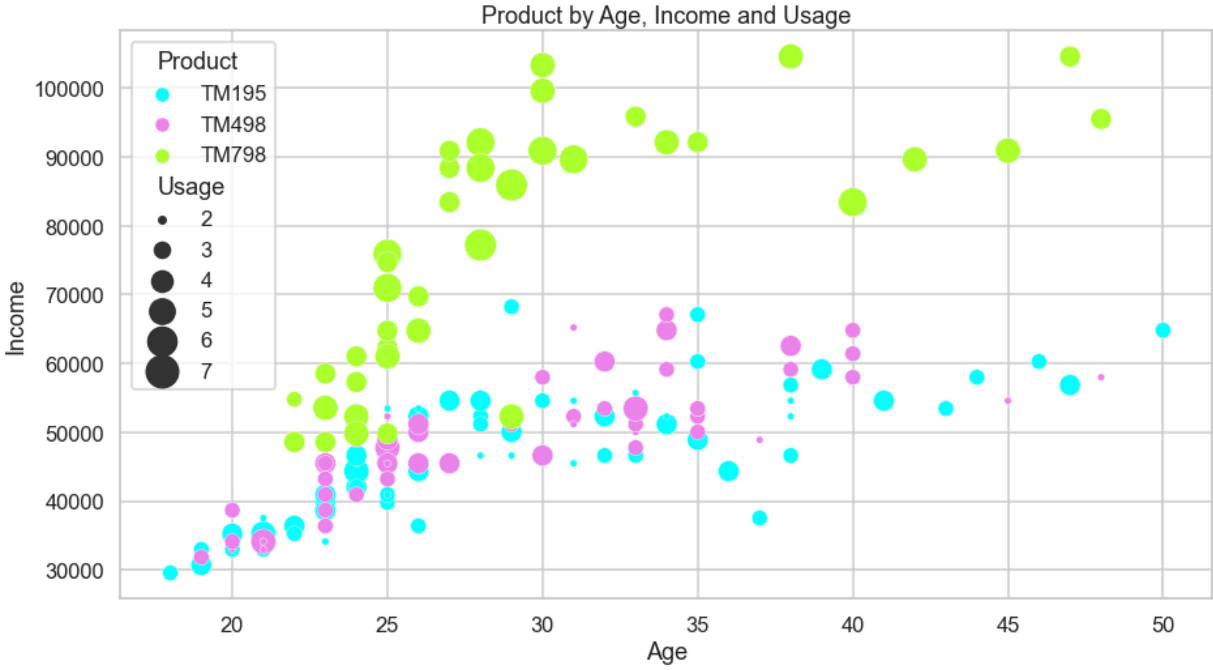
Observations

- The store has 3 models of Treadmill;
- Customers with income greater than $70k tend to buy the product TM798;
- TM195 and TM498 have almost the same customer profile, with income less than 70k.

- **Income vs Age by product and usage**

In [38]:
```
# List of colors to use for the different product
colors = ['cyan', 'violet', 'greenyellow']

# Ploting scatterplot for analysis about correlation between Product, Age, In
plt.figure(figsize=(15,8))

sns.scatterplot(cardio['Age'],cardio['Income'], hue =cardio['Product'],
                size = cardio['Usage'], sizes = (30, 600), palette = colors)

plt.title('Product by Age, Income and Usage')
```

Out[38]: Text(0.5, 1.0, 'Product by Age, Income and Usage')

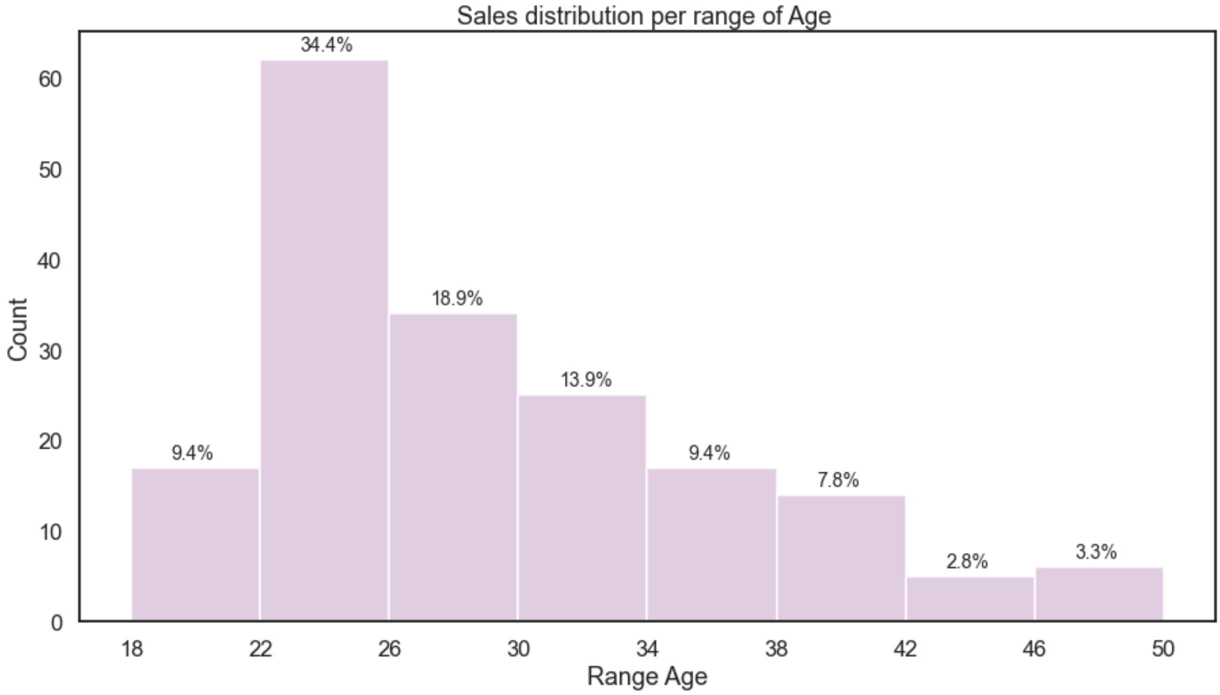


Observations

- TM798 customers and expect to use 5 times per week (average);
- Age concentration for product TM798 and TM195 between 22 to 30 years old;
- TM195 and TM498 have almost the same customer profile, with fitness average of 3.

- **Age concentration**

In [39]:
```
#analyzing age distribution

sns.set_style("white")
plt.figure(figsize=(15,8))

# Plot informations
bins = np.linspace(18,50,9) #creating bins on evenly spaced number on range 1
plot_age = sns.histplot(data = cardio,x='Age',bins=bins, color='thistle')
plt.xlabel('Range Age')
plt.ylabel('Count')
plt.xticks([18, 22, 26, 30, 34, 38, 42, 46, 50]) #specifying ticks via the xt
plt.title('Sales distribution per range of Age ')

# Calculating the length of the column
total = len(cardio['Age'])

# Looping to calculate percentage of bins and annotate the percentage
for age in plot_age.patches:
    percentage = '{:.1f}%'.format(100 * age.get_height()/total)
    x = age.get_x() + age.get_width() /2 -0.75
    y = age.get_y() + age.get_height() + 1
    plot_age.annotate(percentage, (x, y), size = 14)
```

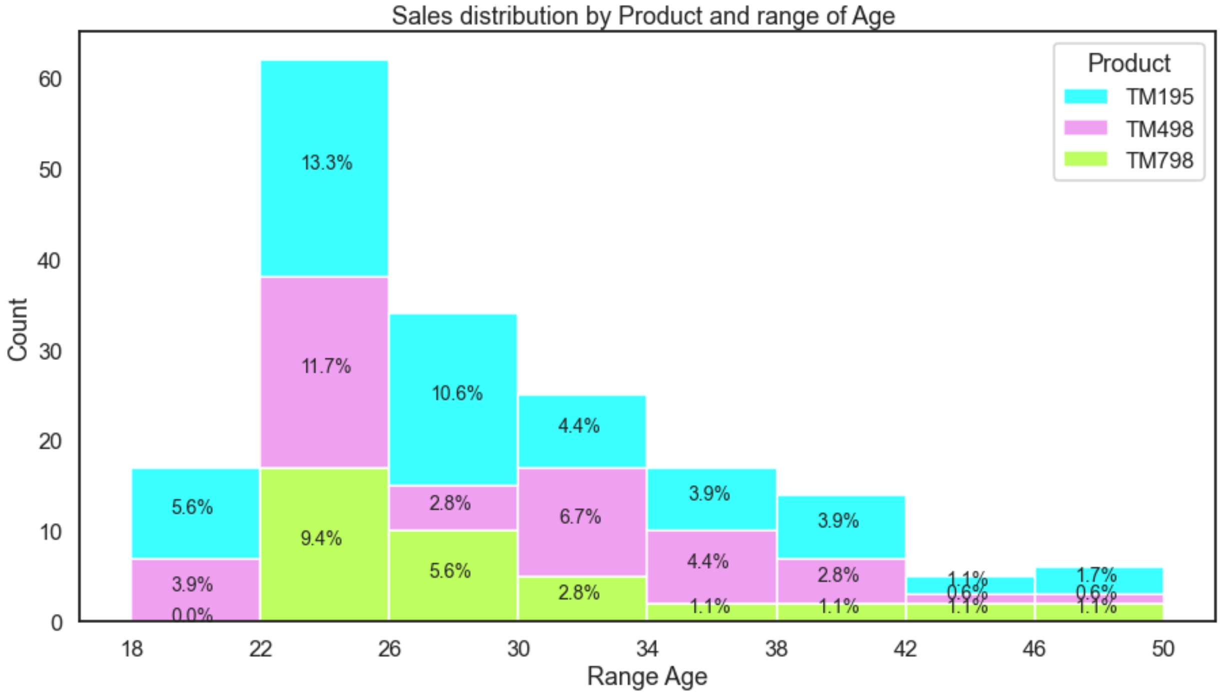


Observation

- 67% of sales are concentrated on customer with range of age between 22 to 33 years old.

- **Age distribution by product**

In [40]:
```
#analyzing age distribution for product

sns.set_style("white")
plt.figure(figsize=(15,8))

# Plot informations
colors = ['cyan', 'violet', 'greenyellow']
bins = np.linspace(18,50,9) #creating bins on evenly spaced number on range 1

# Plotting stacked histogram
plot_age = sns.histplot(data = cardio,x='Age',bins=bins, hue="Product", multi
plt.xlabel('Range Age')
plt.ylabel('Count')
plt.xticks([18, 22, 26, 30, 34, 38, 42, 46, 50]) #specifying ticks via the xt
plt.title('Sales distribution by Product and range of Age ')

# Calculating the length of the column
total = len(cardio['Age'])

# Looping to calculate percentage of bins and annotate the percentage
for age in plot_age.patches:
    percentage = '{:.1f}%'.format(100 * age.get_height()/total)
    x = age.get_x() + age.get_width() /2 -0.75
    y = age.get_y() + age.get_height() /2
    plot_age.annotate(percentage, (x, y), size = 14)
```



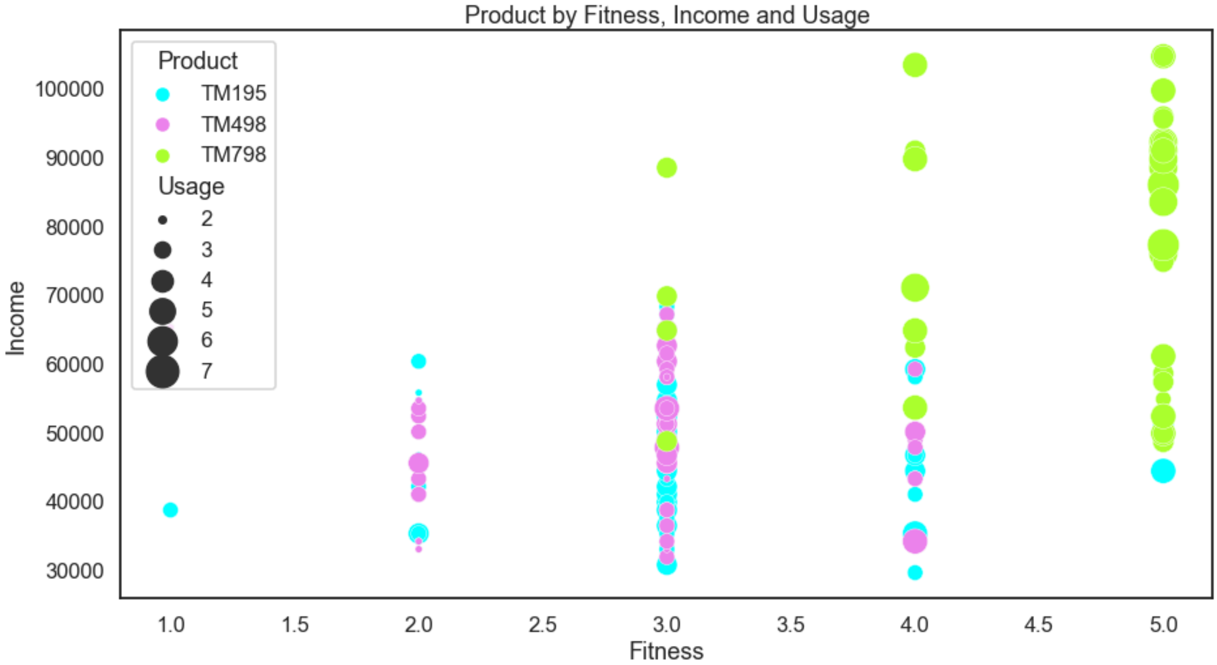
Observation

- TM195 Range 18 to 29 years old represents 30% of total sales;
- TM498 Has a mixed range, 22 to 25 and 30 to 37, 23% of sales;
- TM798 Range 22 to 33 years old, 18% of total sales.

- **Income vs Fitness by product and usage**

In [41]:
```
# List of colors to use for the different product
colors = ['cyan', 'violet', 'greenyellow']

# Ploting scatterplot for analysis about correlation between Product, Age, In
plt.figure(figsize=(15,8))

sns.scatterplot(cardio['Fitness'],cardio['Income'], hue =cardio['Product'],
                size = cardio['Usage'], sizes = (30, 600), palette = colors)

plt.title('Product by Fitness, Income and Usage')
```

Out[41]:
```
Text(0.5, 1.0, 'Product by Fitness, Income and Usage')
```



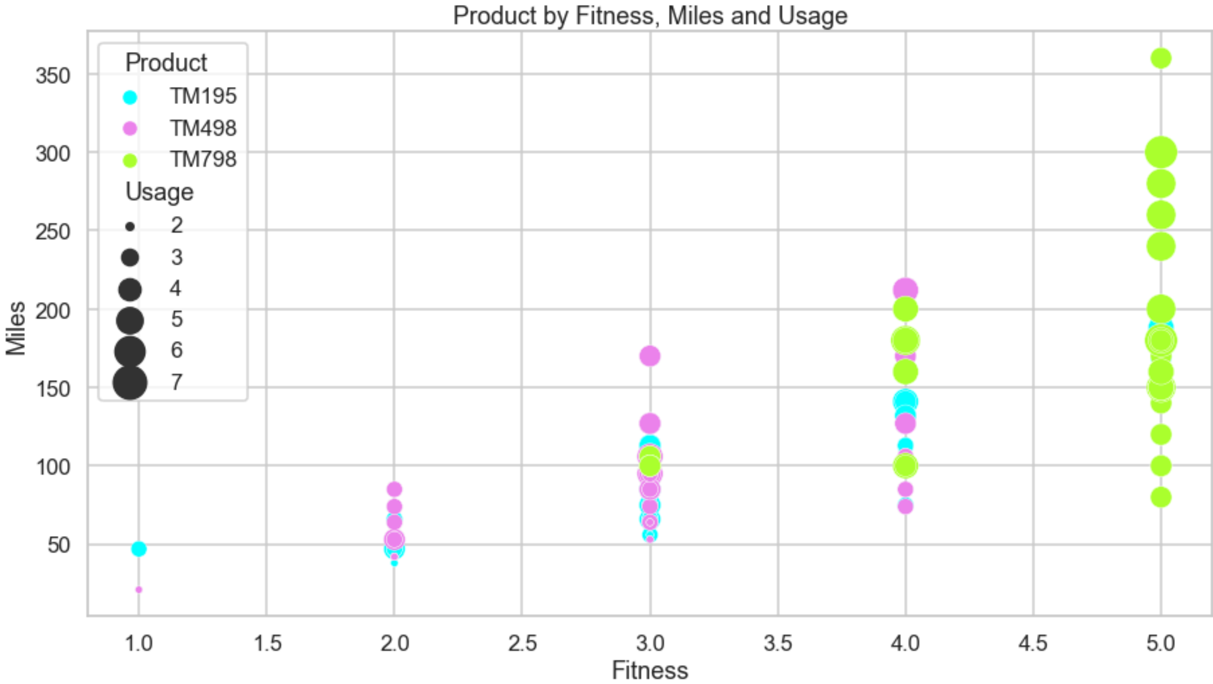
Observation

- TM798 customer self rate themself between 4 and 5;
- There is a TM195 customer that self rate itself as 1.
- Higher income has strong correlation with greater fitness.

- **Fitness vs Miles by product**

In [42]:
```
# List of colors to use for the different product
colors = ['cyan', 'violet', 'greenyellow']

# Ploting scatterplot for analysis about correlation between Product, Fitness
sns.set_style("whitegrid")
plt.figure(figsize=(15,8))

sns.scatterplot(x='Fitness', y='Miles', data=cardio, hue=cardio['Product'],si
                sizes = (30, 600),palette = colors)

plt.title('Product by Fitness, Miles and Usage')
```

Out[42]: `Text(0.5, 1.0, 'Product by Fitness, Miles and Usage')`



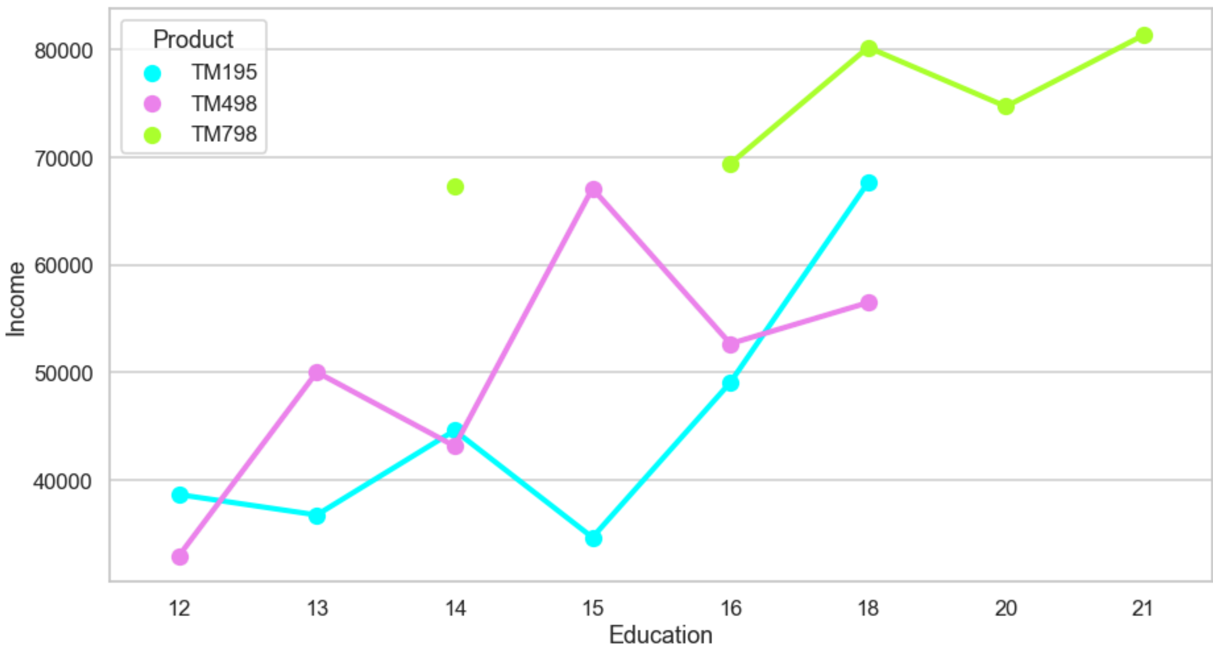
Observation

- Fitness and Miles has a hight strong correlation
- The product TM798 is used for people that is verry fit and expected to run more

- **Income vs Education by product**

In [43]:
```
# List of colors to use for the different product
colors = ['cyan', 'violet', 'greenyellow']

# Ploting scatterplot for analysis about correlation between Product, Age, In
plt.figure(figsize=(15,8))

sns.pointplot(x='Education', y='Income', data=cardio, hue='Product', ci=None,
```

Out[43]: `<AxesSubplot:xlabel='Education', ylabel='Income'>`



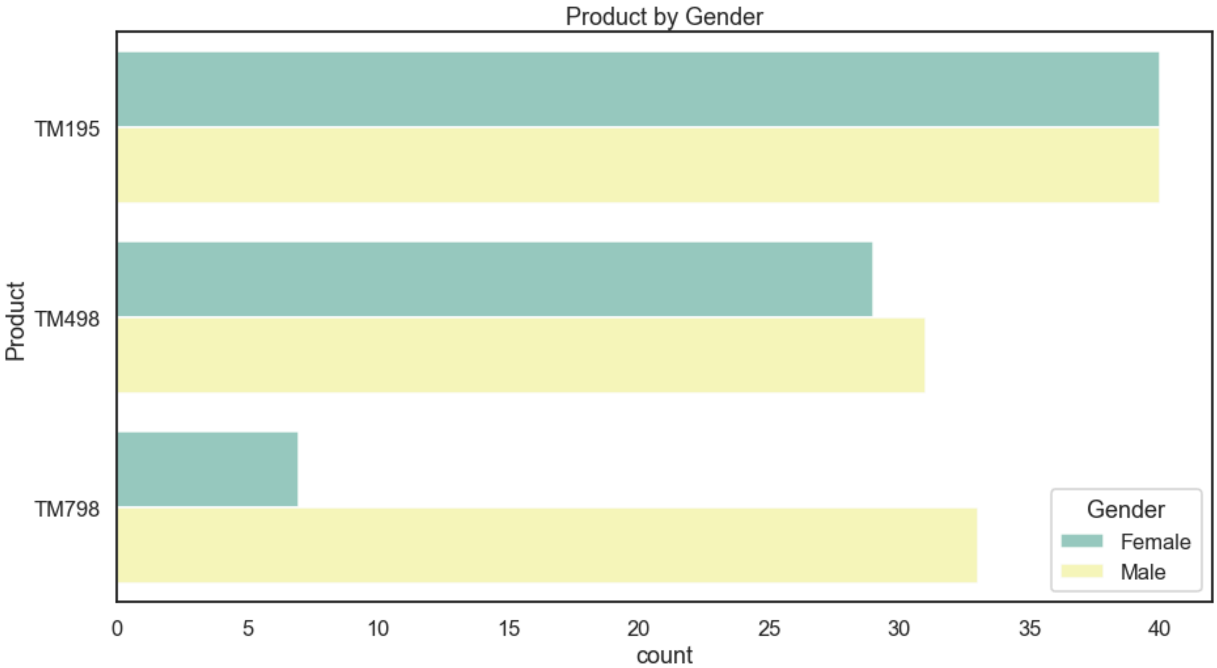
Observation

- Income and Education has some correlation
- Customer with more years of education tend to buy TM798.

- **Product vs Gender**

In [44]:
```
# Ploting countplot for analysis between Product and Gender
sns.set_style("white")
plt.figure(figsize=(15,8))

sns.countplot(y=cardio['Product'], hue = cardio['Gender'], palette="Set3")

plt.title('Product by Gender')
```

Out[44]: `Text(0.5, 1.0, 'Product by Gender')`



- **Product vs Fitness by Gender**

In [45]:
```
# Ploting swarmplot for analysis between Product, Fitness and Gender
sns.set_style("dark")
plt.figure(figsize=(15,8))

sns.swarmplot(x='Product', y='Fitness', data=cardio, hue=cardio['Gender'], si

plt.title('Product by Fitness and Gender')
```

Out[45]: `Text(0.5, 1.0, 'Product by Fitness and Gender')`

In [46]:
```
gender_dist_TM498 = cardio[cardio['Product'] == "TM498"]
gender_dist_TM498['Gender'].value_counts()
```

Out[46]:
```
Male      31
Female    29
Name: Gender, dtype: int64
```

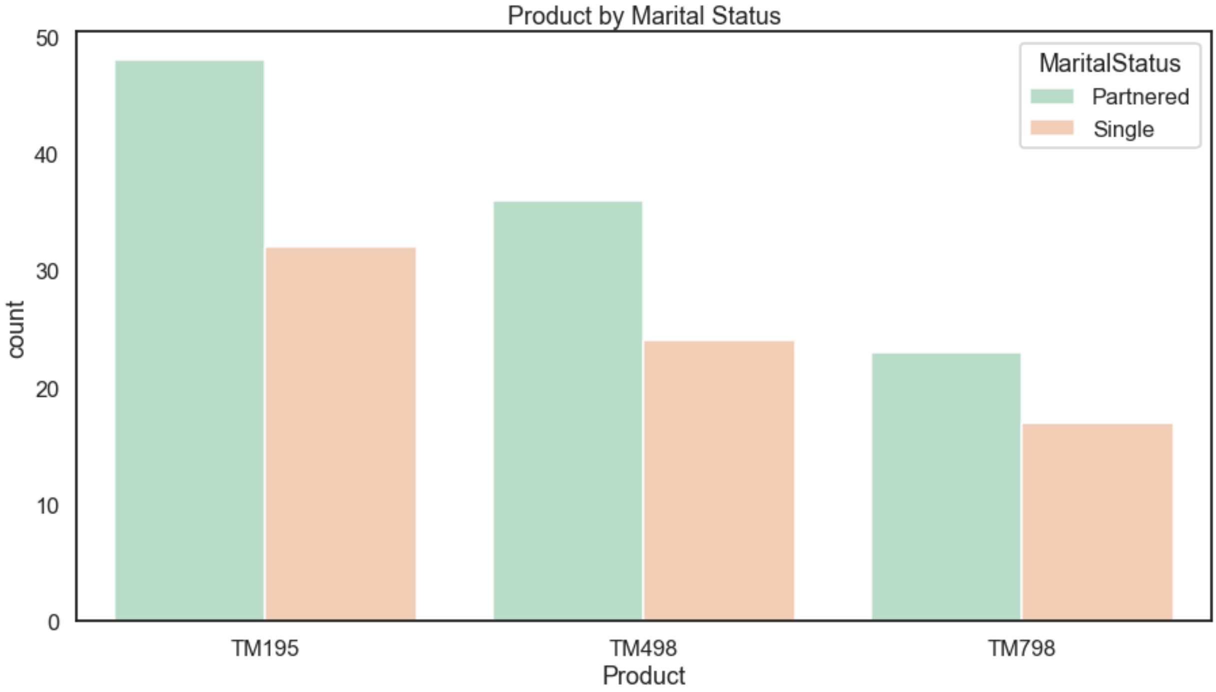
In [47]:
```
gender_dist_TM798 = cardio[cardio['Product'] == "TM798"]
gender_dist_TM798['Gender'].value_counts()
```

Out[47]:
```
Male      33
Female     7
Name: Gender, dtype: int64
```

- **Product by Marital Status**

In [48]:
```
# Ploting countplot for analysis between Product and Marital Status
sns.set_style("white")
plt.figure(figsize=(15,8))

sns.countplot(x=cardio['Product'], hue = cardio['MaritalStatus'], palette="Pa

plt.title('Product by Marital Status')
```

Out[48]:
```
Text(0.5, 1.0, 'Product by Marital Status')
```



Observation

- TM195 has equal number of Female and Male customers (40) and higher number of partnered.
- TM498 sales for male (31) is a little bit higher than for Female (29) and has higher number of partnered.
- TM798 sales for male (33) is considerably higher than for Female (7) , also preferable for partnered customers and the ones that self rate equal or greater than 3.