Business Problem:

The market research team at AeroFit wants to identify the characteristics of the target audience for each type of treadmill offered by the company, to provide a better recommendation of the treadmills to the new customers. The team decides to investigate whether there are differences across the product with respect to customer characteristics.

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:

```
#Importing datasets
aerofit_df = pd.read_csv("aerofit_treadmill.csv")
```

In [3]:

```
#exploring the dataset
aerofit_df.head()
```

Out[3]:

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

In [4]:

```
#Data Shapes
aerofit_df.shape
```

Out[4]:

```
(180, 9)
```

In [10]:

```
print("Number of rows in aerofit dataset: "+ str(aerofit_df.shape[0]))
print("Number of columns in aerofit dataset: "+ str(aerofit_df.shape[1]))
```

```
Number of rows in aerofit dataset: 180
Number of columns in aerofit dataset: 9
```

In [11]:

```
#Looking into datatypes
aerofit_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180 entries, 0 to 179
Data columns (total 9 columns):
Product          180 non-null object
Age              180 non-null int64
Gender           180 non-null object
Education        180 non-null int64
MaritalStatus    180 non-null object
Usage            180 non-null int64
Fitness          180 non-null int64
Income           180 non-null int64
Miles            180 non-null int64
dtypes: int64(6), object(3)
memory usage: 12.8+ KB
```

In [18]:

```
#changing it to  object dtype to category  to save memory
aerofit_df.Product=aerofit_df["Product"].astype("category")
aerofit_df.Gender=aerofit_df["Gender"].astype("category")
aerofit_df.MaritalStatus=aerofit_df["MaritalStatus"].astype("category")
```

In [20]:

```
sns.set_style("whitegrid")
```

Outliers Detection

In [44]:

```
outliers = []
def detect_outliers_iqr(data):
    data = sorted(data)
    q1 = np.percentile(data, 25)
    q3 = np.percentile(data, 75)
    # print(q1, q3)
    IQR = q3-q1
    lwr_bound = q1-(1.5*IQR)
    upr_bound = q3+(1.5*IQR)
    # print(lwr_bound, upr_bound)
    for i in data:
        if (i<lwr_bound or i>upr_bound):
            outliers.append(i)
    return outliers# Driver code
```

In [41]:

```
numeric_cols = []
for i in aerofit_df.columns:
    if(aerofit_df[i].dtype == "int64"):
        numeric_cols.append(i)
```

In [42]:

```
print(numeric_cols)
```

```
['Age', 'Education', 'Usage', 'Fitness', 'Income', 'Miles']
```

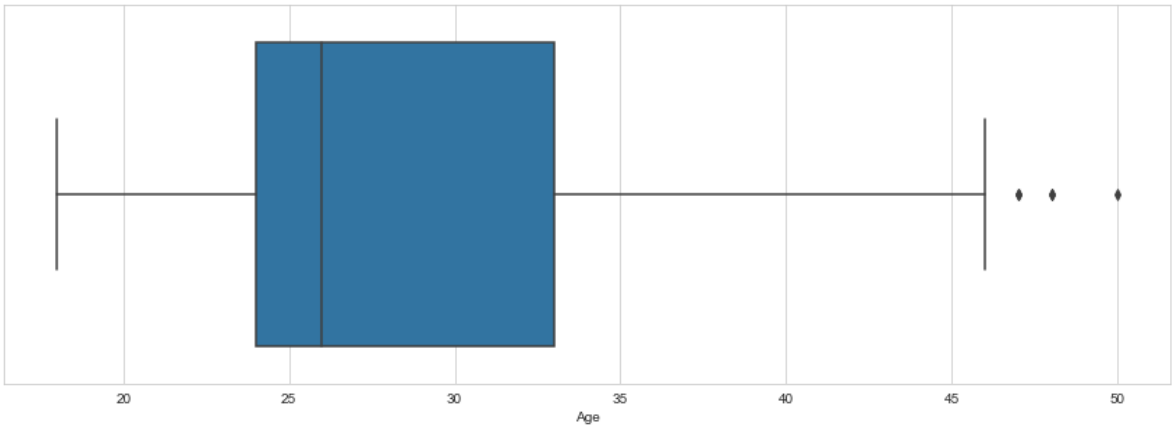
In [107]:

```
def detect_outliers_iqr(data):
    outliers = []
    data = sorted(data)
    q1 = np.percentile(data, 25)
    q3 = np.percentile(data, 75)
    print('q1 and q3 : ', q1, q3)
    IQR = q3-q1
    lwr_bound = q1-(1.5*IQR)
    upr_bound = q3+(1.5*IQR)
    print('lower bound : {} and higher bound : {}'.format(lwr_bound, upr_bound))
    for i in data:
        if (i<lwr_bound or i>upr_bound):
            outliers.append(i)
    return outliers# Driver code
```

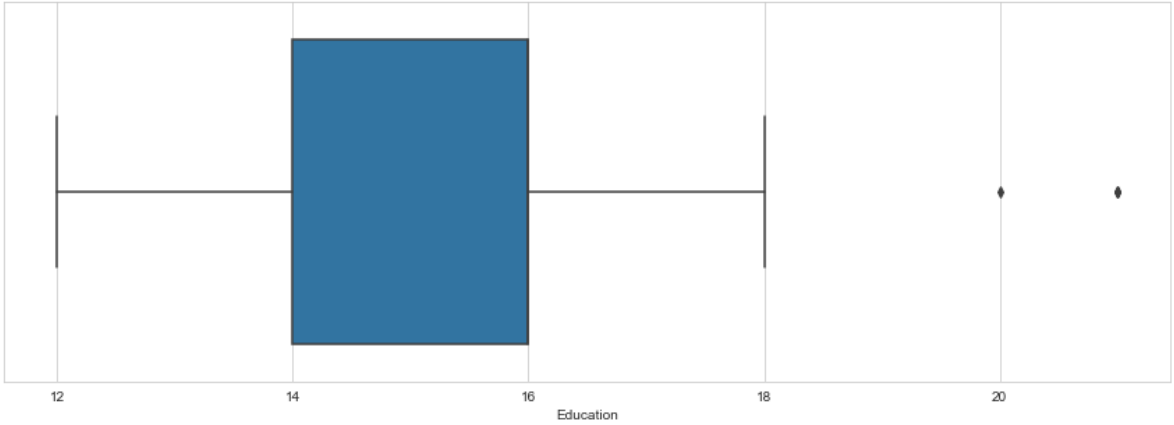
In [108]:

```
# row = len(numeric_cols)//2
# col = len(numeric_cols)//row
# ax_cnt = 1

for i in numeric_cols:

    outliers = detect_outliers_iqr(aerofit_df[i])
    outliers_n = len(outliers)
    print("{} Outliers from {} column using IQR method: {}" .format(outliers_n,i, outliers)
    print("----------------------------------------------------------------------")
    print("----------------------------------------------------------------------")
    #plt.subplot(row, col, ax_cnt)
    fig = plt.figure(figsize=(15,5))
    sns.boxplot(x = i, data = aerofit_df)
    col_cnt = col_cnt + 1
    plt.show()
```

```
q1 and q3 :  24.0 33.0
lower bound : 10.5 and higher bound : 46.5
5 Outliers from Age column using IQR method: [47, 47, 48, 48, 50]
----------------------------------------------------------------------
----------------------------------------------------------------------
```

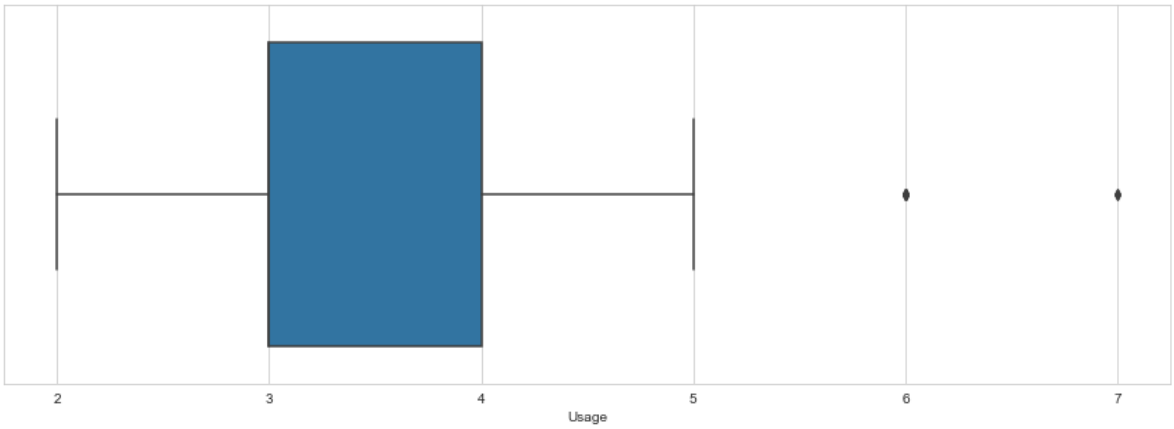



```
q1 and q3 :  14.0 16.0
lower bound : 11.0 and higher bound : 19.0
4 Outliers from Education column using IQR method: [20, 21, 21, 21]
----------------------------------------------------------------------
----------------------------------------------------------------------
```

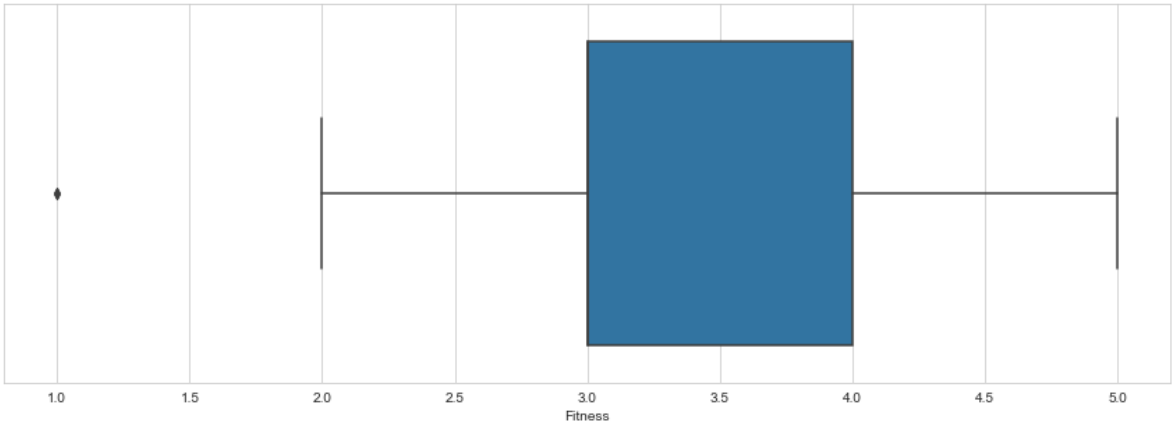



```
q1 and q3 :  3.0 4.0
lower bound : 1.5 and higher bound : 5.5
```

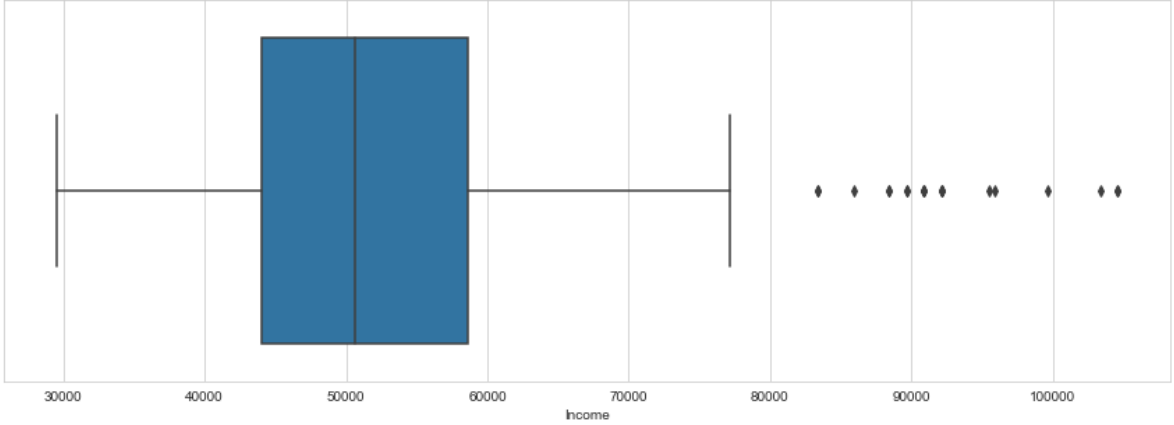
```
9 Outliers from Usage column using IQR method: [6, 6, 6, 6, 6, 6, 6, 7, 7]
-----------------------------------------------------------------
-----------------------------------------------------------------
```




```
q1 and q3 :  3.0 4.0
lower bound : 1.5 and higher bound : 5.5
2 Outliers from Fitness column using IQR method: [1, 1]
-----------------------------------------------------------------
-----------------------------------------------------------------
```

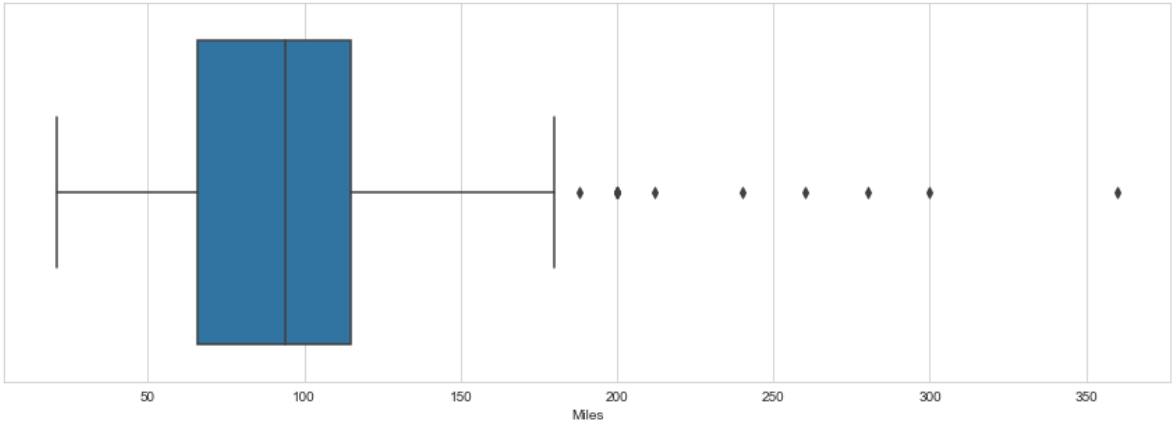



```
q1 and q3 :  44058.75 58668.0
lower bound : 22144.875 and higher bound : 80581.875
19 Outliers from Income column using IQR method: [83416, 83416, 85906, 8839
6, 88396, 89641, 89641, 90886, 90886, 90886, 92131, 92131, 92131, 95508, 958
66, 99601, 103336, 104581, 104581]
-----------------------------------------------------------------
-----------------------------------------------------------------
```




```
q1 and q3 :  66.0 114.75
lower bound : -7.125 and higher bound : 187.875
13 Outliers from Miles column using IQR method: [188, 200, 200, 200, 200,
200, 200, 212, 240, 260, 280, 300, 360]
```

```
------------------------------------------------------------------
------------------------------------------------------------------
```



Missing data Checks

In [121]:

```
aerofit_df.isnull().sum()
```

Out[121]:

```
Product          0
Age              0
Gender           0
Education        0
MaritalStatus    0
Usage            0
Fitness          0
Income           0
Miles            0
dtype: int64
```

There is no missing data in the datasets

Data Integrity Checks

In [124]:

```
for i in aerofit_df.columns:
    print(aerofit_df[i].unique())
    print("=========================================")
```

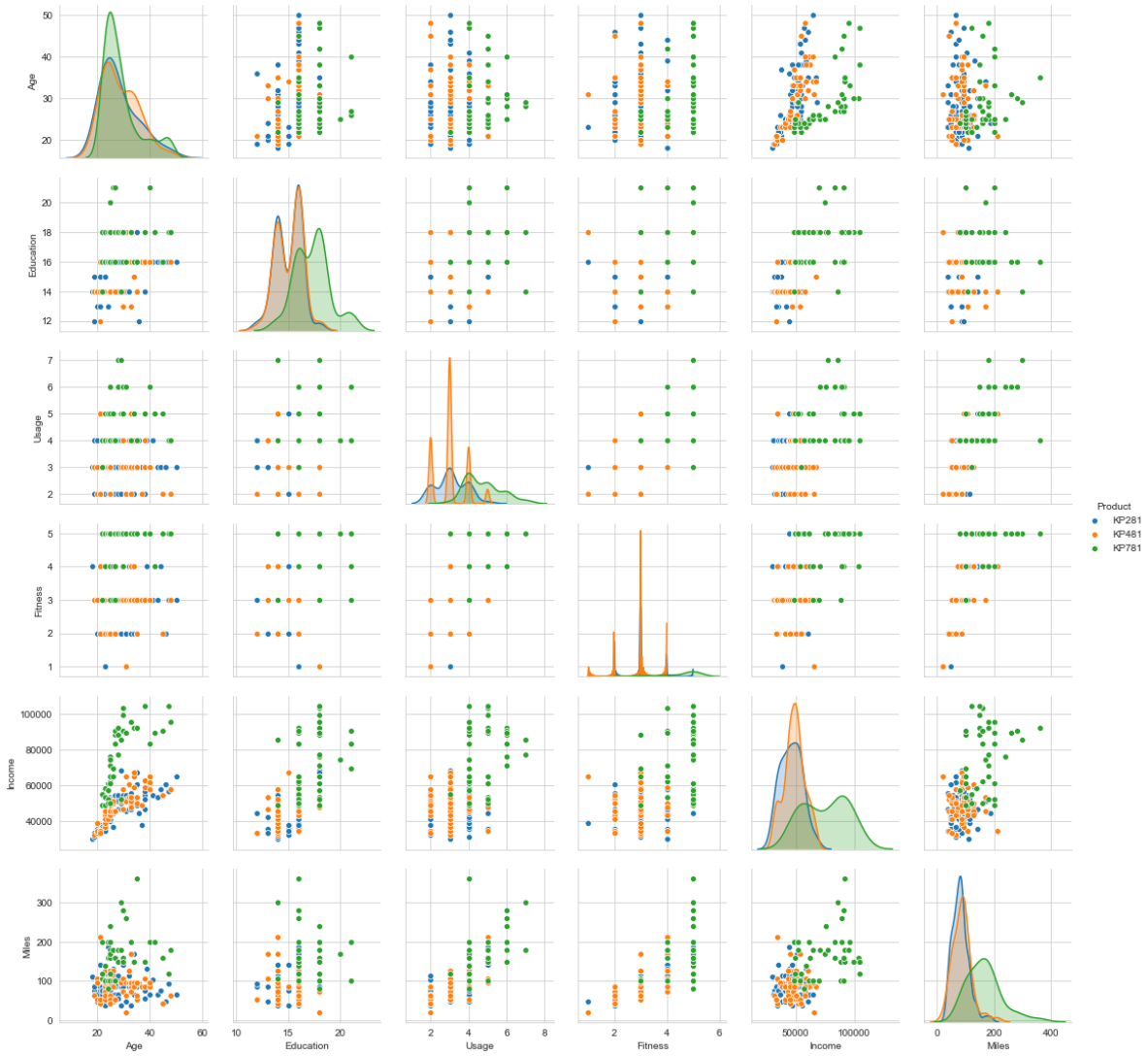
```
[KP281, KP481, KP781]
Categories (3, object): [KP281, KP481, KP781]
=========================================
[18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41
 43 44 46 47 50 45 48 42]
=========================================
[Male, Female]
Categories (2, object): [Male, Female]
=========================================
[14 15 12 13 16 18 20 21]
=========================================
[Single, Partnered]
Categories (2, object): [Single, Partnered]
=========================================
[3 2 4 5 6 7]
=========================================
[4 3 2 1 5]
=========================================
[ 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]
=========================================
[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]
=========================================
```

There is misplaced data

Correlation

In [166]:

```
#Lets look at the overall picture and Correlation
sns.pairplot(aerofit_df, hue = 'Product')
plt.show()
```

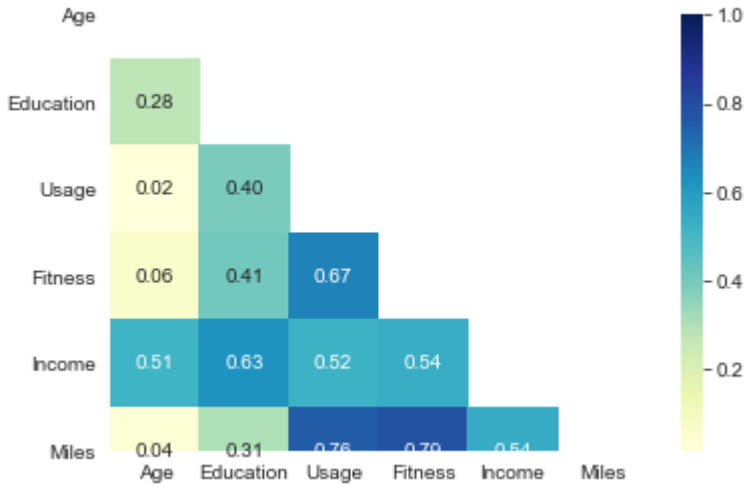


In [168]:

```
corr = aerofit_df.corr()
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)] = True
corr = (corr)
sns.heatmap(corr,
            annot = True,
            fmt = ".2f",
            mask = mask,
            xticklabels=corr.columns.values,
            yticklabels=corr.columns.values,
            cmap="YlGnBu")

corr
```

Out[168]:

	Age	Education	Usage	Fitness	Income	Miles
Age	1.000000	0.280496	0.015064	0.061105	0.513414	0.036618
Education	0.280496	1.000000	0.395155	0.410581	0.625827	0.307284
Usage	0.015064	0.395155	1.000000	0.668606	0.519537	0.759130
Fitness	0.061105	0.410581	0.668606	1.000000	0.535005	0.785702
Income	0.513414	0.625827	0.519537	0.535005	1.000000	0.543473
Miles	0.036618	0.307284	0.759130	0.785702	0.543473	1.000000



Insights:

Moderate Correlation:

- Age and Income has correlation : 0.51
- Income and Usage have correlation : 0.52

- Miles and Income have correlation : 0.54
- Fitness and Income have correlation : 0.54

High Correlation

- Education and Income have correlation : 0.63
- Fitness and Usage have correlation : 0.67
- Usage and Miles have high correlation : 0.75
- Miles and Fitness have high correlation : 0.79

Product Usage

In [127]:

```
sns.countplot(x='Product', data = aerofit_df)
plt.title("Product Purchased by Customers")
plt.show()
```

In [134]:

```
aerofit_df[aerofit_df['Product']=='KP281'].describe().T
```

Out[134]:

	count	mean	std	min	25%	50%	75%	max
Age	80.0	28.5500	7.221452	18.0	23.0	26.0	33.0	50.0
Education	80.0	15.0375	1.216383	12.0	14.0	16.0	16.0	18.0
Usage	80.0	3.0875	0.782624	2.0	3.0	3.0	4.0	5.0
Fitness	80.0	2.9625	0.664540	1.0	3.0	3.0	3.0	5.0
Income	80.0	46418.0250	9075.783190	29562.0	38658.0	46617.0	53439.0	68220.0
Miles	80.0	82.7875	28.874102	38.0	66.0	85.0	94.0	188.0

Insights

- 80 customers have purchased KP281 Model.

- The average age of customers who bought KP281 is 28.5, whereas the min and max age is 18 and 50 respectively.
- Average Education is 15 and median is 16.
- Expected usage is 3 day a week

In [135]:

```
aerofit_df[aerofit_df['Product']=='KP481'].describe().T
```

Out[135]:

	count	mean	std	min	25%	50%	75%	max
Age	60.0	28.900000	6.645248	19.0	24.0	26.0	33.25	48.0
Education	60.0	15.116667	1.222552	12.0	14.0	16.0	16.00	18.0
Usage	60.0	3.066667	0.799717	2.0	3.0	3.0	3.25	5.0
Fitness	60.0	2.900000	0.629770	1.0	3.0	3.0	3.00	4.0
Income	60.0	48973.650000	8653.989388	31836.0	44911.5	49459.5	53439.00	67083.0
Miles	60.0	87.933333	33.263135	21.0	64.0	85.0	106.00	212.0

Insights

- 60 customers have purchased KP281 Model.
- The average age of customers who bought KP481 is 28.9, whereas the min and max age is 19 and 48 respectively.
- Average Education is 15 and median is 16.
- Expected usage is 3 day a week

In [136]:

```
aerofit_df[aerofit_df['Product']=='KP781'].describe().T
```

Out[136]:

	count	mean	std	min	25%	50%	75%	max
Age	40.0	29.100	6.971738	22.0	24.75	27.0	30.25	48.0
Education	40.0	17.325	1.639066	14.0	16.00	18.0	18.00	21.0
Usage	40.0	4.775	0.946993	3.0	4.00	5.0	5.00	7.0
Fitness	40.0	4.625	0.667467	3.0	4.00	5.0	5.00	5.0
Income	40.0	75441.575	18505.836720	48556.0	58204.75	76568.5	90886.00	104581.0
Miles	40.0	166.900	60.066544	80.0	120.00	160.0	200.00	360.0

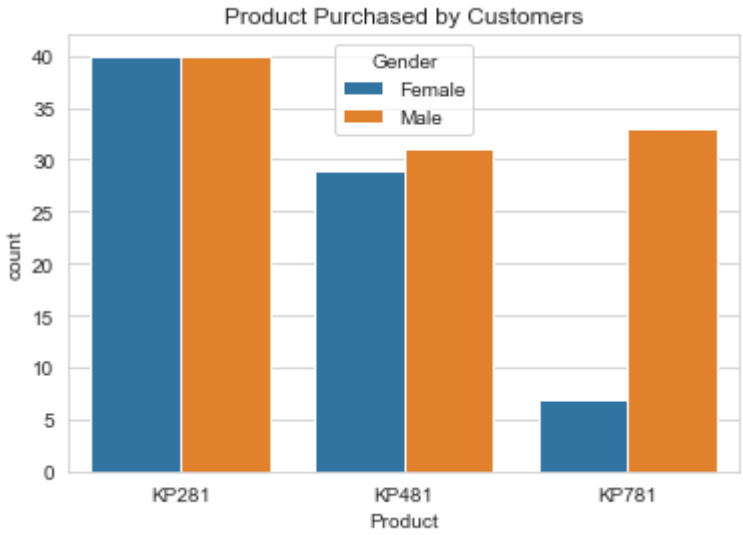
Insights

- 40 customers have purchased KP281 Model.
- The average age of customers who bought KP481 is 29, whereas the min and max age is 22 and 48 respectively.
- Average Education is 17 and median is 18.
- Expected usage is 5 day a week

KP781 Models are mostly puchased by those customers who earns more and have better fitness and use the product more frequently

In [146]:

```
sns.countplot(x='Product', hue = 'Gender', data = aerofit_df)
plt.title("Product Purchased by Customers")
plt.show()
```



Female customers are prefering KP281 and KP481 compared to KP781

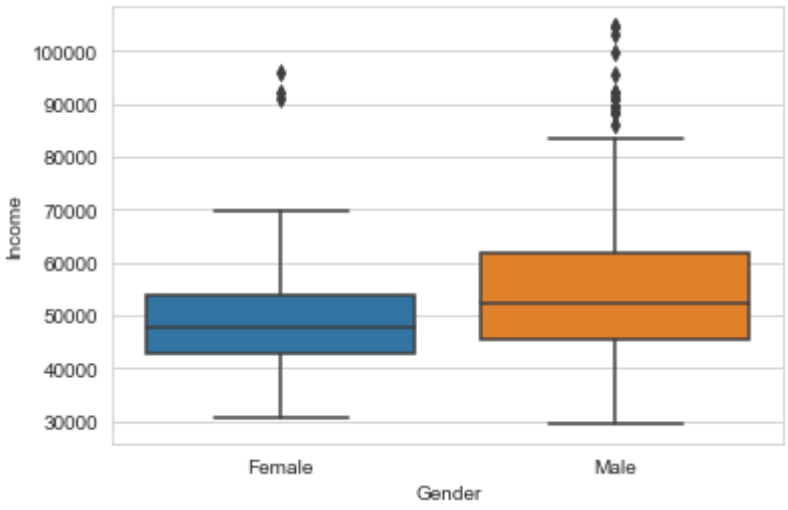
Gender vs Income

In [138]:

```
sns.boxplot(x='Gender', y='Income', data= aerofit_df)
```

Out[138]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x153943bdf48>
```



Median Salary is higher for MALE compared to FEMALE customers

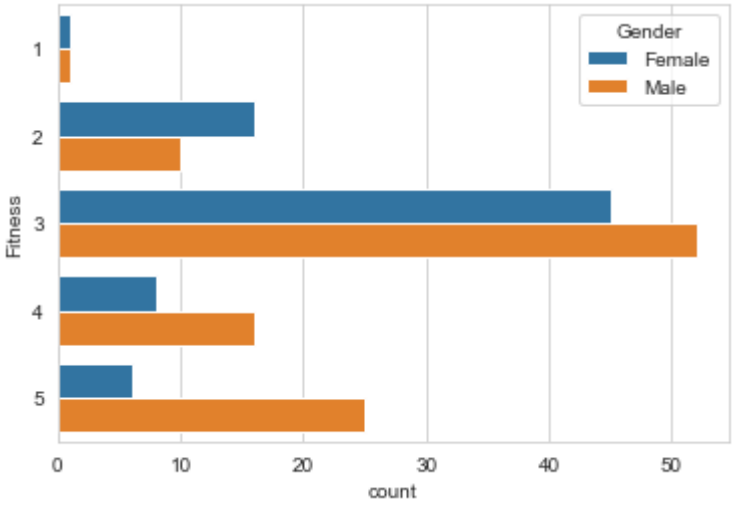
Gender vs Fitness

In [144]:

```
sns.countplot(hue='Gender', y='Fitness', data= aerofit_df)
```

Out[144]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x15391eee248>
```



- In the low fitness rating Female customers have more consentration than Male, based on the above chart Male customers are more fit compared to Female

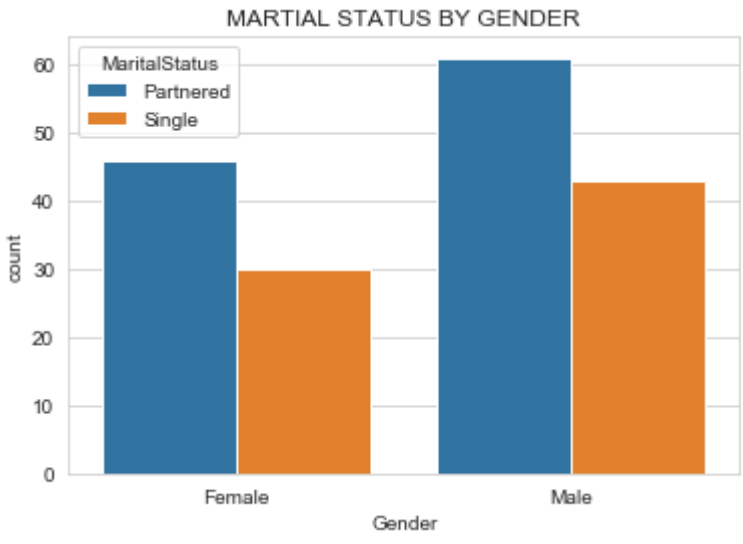
Gender vs Marital Status

In [145]:

```
sns.countplot(aerofit_df['Gender'],hue=aerofit_df["MaritalStatus"]).set(title='MARTIAL STAT
```

Out[145]:

```
[Text(0.5, 1.0, 'MARTIAL STATUS BY GENDER')]
```



- Overall Partnered people have puchased more product than Single customers.

- Partnerd Male customers have bought more products than female counter part

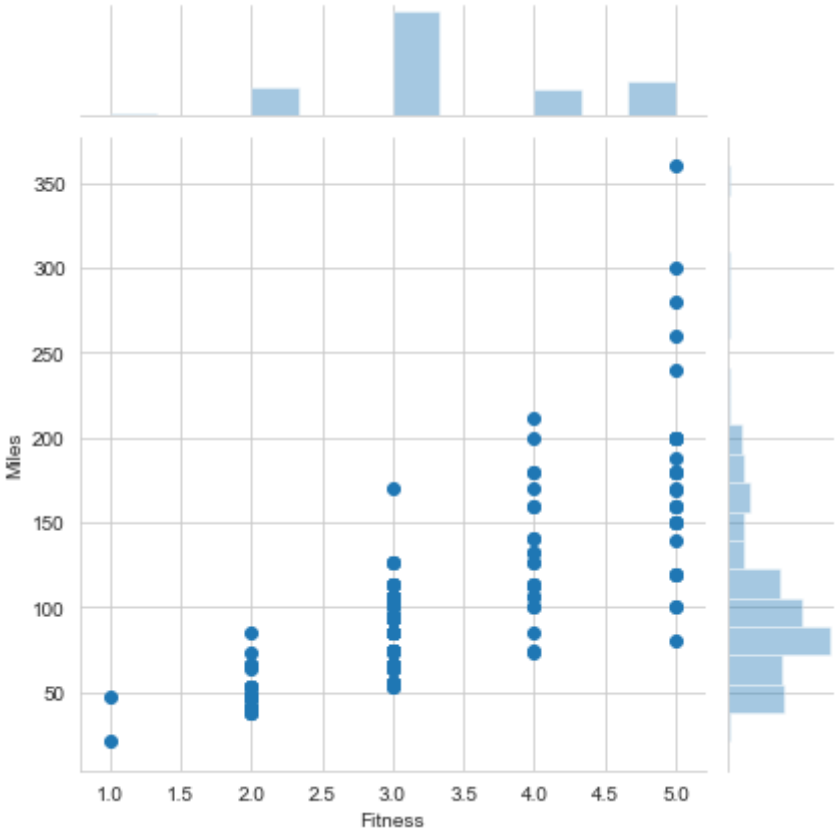
Miles vs Fitness

In [165]:

```
sns.jointplot(data=aerofit_df, x="Fitness", y="Miles")
```

Out[165]:

```
<seaborn.axisgrid.JointGrid at 0x1539400e548>
```

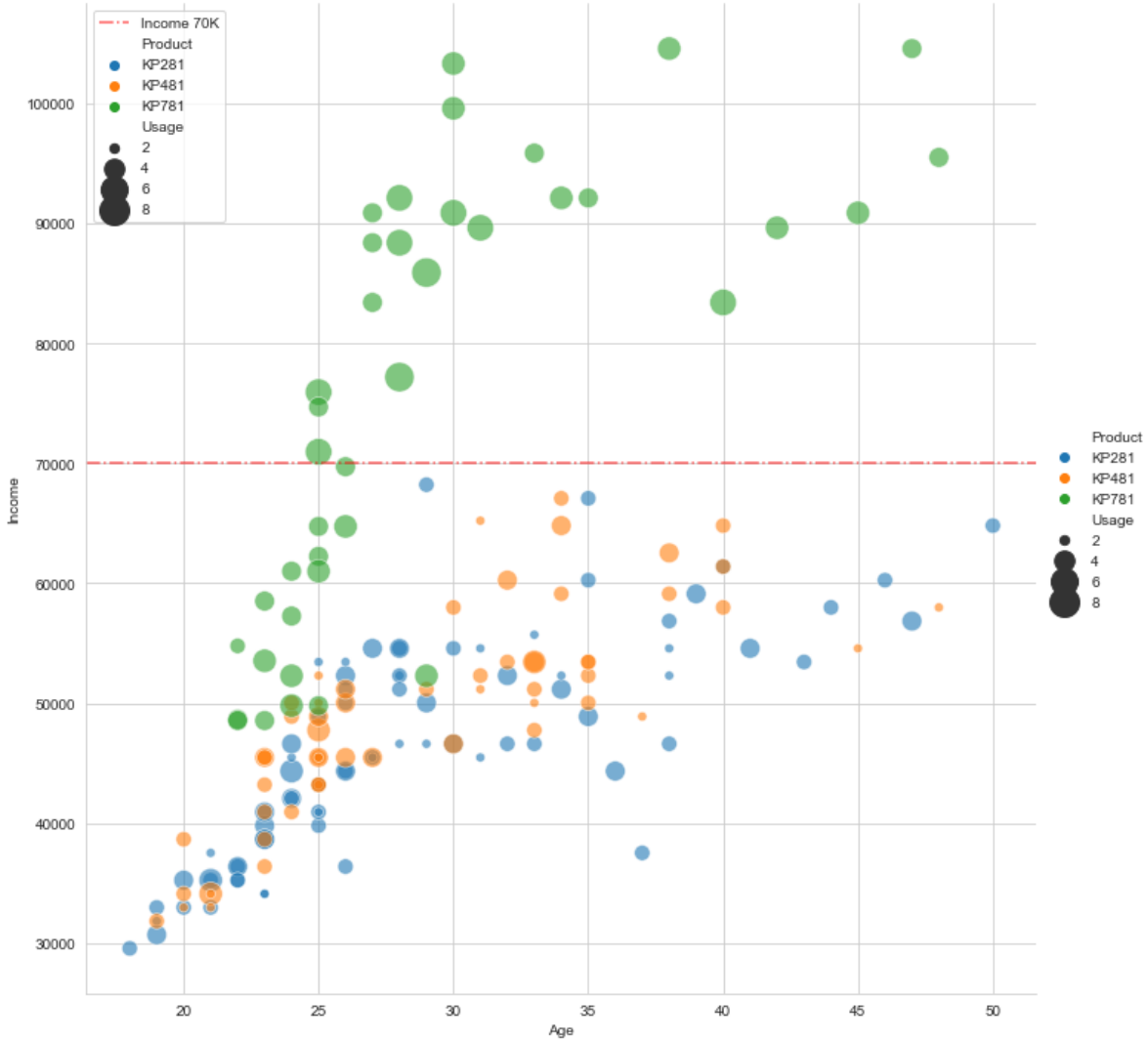


- Fitness has increased with more Miles , which means customers who have walked more Miles have a tendency to much more fit than others
- Most of the people have walked between 40 to 125 miles.

Age vs Income vs Product vs Usage

In [158]:

```
sns.relplot(x='Age', y='Income', hue='Product', size = "Usage",sizes=(40, 400), alpha=.6,
            height=10, data=aerofit_df)
plt.axhline(70000,alpha = 0.6,  linestyle='-.', color='r', label='Income 70K')
plt.legend()
plt.show()
```



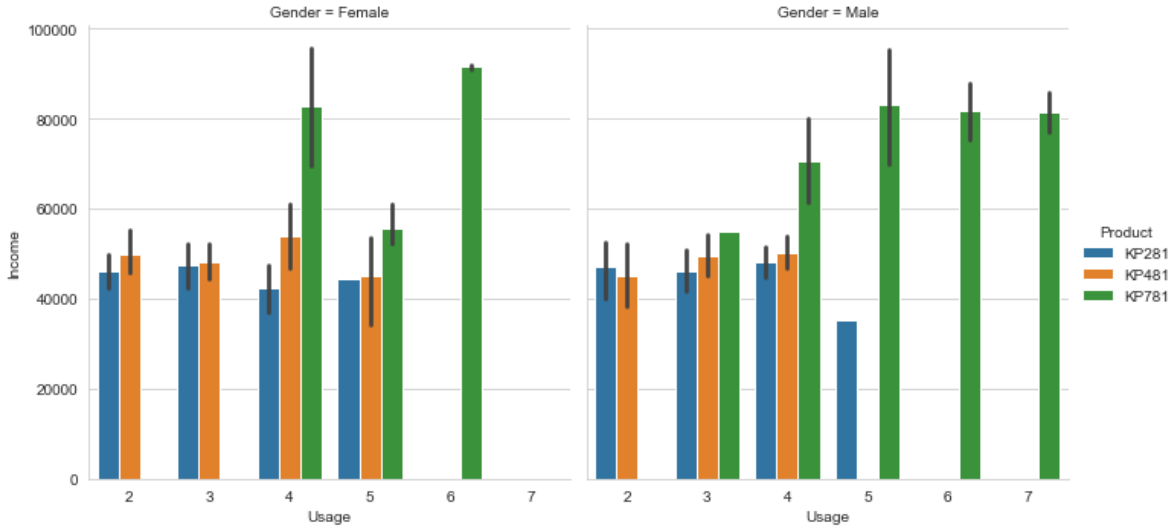
- For customers with Age 22-30, Income has increased significantly and most of them have choosen KP781
- Customers with Age between 22 to 30 with income more than 70000 are mostly purchasing the KP781 model and their usage are higher than the other customers with other two models.

Usage vs Income vs Product vs Gender

In [154]:

```
plt.figure(figsize=(14,7))
sns.catplot(x='Usage', y='Income', col='Gender',hue='Product' ,kind="bar", data=aerofit_df)
plt.show()
```

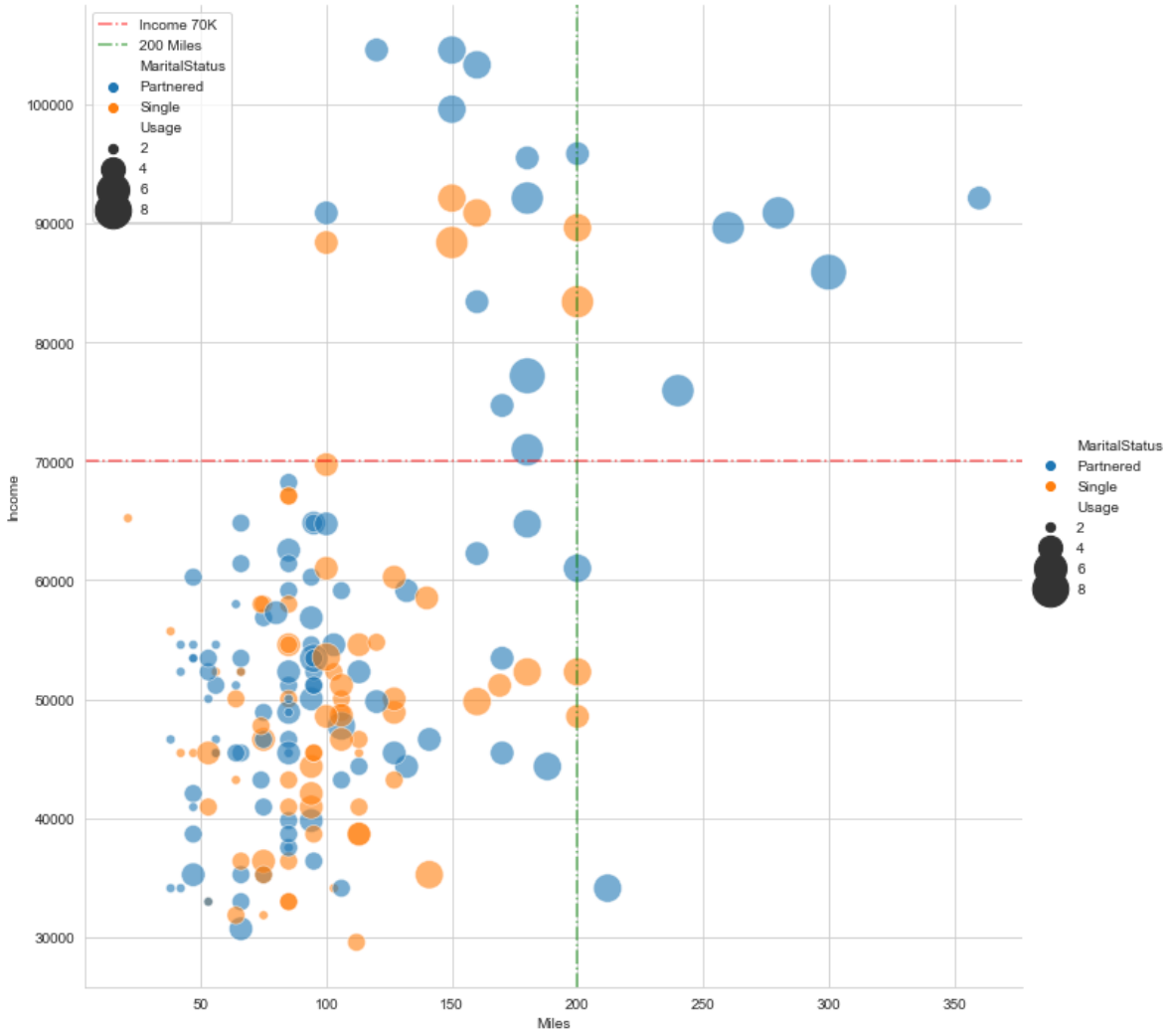
```
<Figure size 1008x504 with 0 Axes>
```



Miles vs Income vs Marital Status vs Fitness

In [162]:

```
sns.relplot(x='Miles', y='Income', hue='MaritalStatus', size = "Usage",sizes=(40, 600), alp
            height=10, data=aerofit_df)
plt.axhline(70000,alpha = 0.6,  linestyle='-.', color='r', label='Income 70K')
plt.axvline(200,alpha = 0.6,  linestyle='-.', color='g', label='200 Miles')
plt.legend()
plt.show()
```



Customer Profiling:

Binning - Age and Income

In [193]:

```
bins = [10,20,30,40,60]
labels =["Teens","20s","30s","Above 40s"]
aerofit_df['Age_Groups'] = pd.cut(aerofit_df['Age'], bins)
aerofit_df['Age_Groups_Code'] = pd.cut(aerofit_df['Age'], bins,labels=labels)
```

In [194]:

```
bins_income = [20000, 30000,  50000, 70000, 10000000]
labels_income = ['Low Income Group','Lower-Middle Income Group','Upper-Middle Income Group'
aerofit_df['Income_Groups'] = pd.cut(aerofit_df['Income'],bins_income,labels = labels_incom
```

In [195]:

```
bins_income = [10 , 13,  17, 50]
labels_education = ['Low_Educational_Qualification','Average_Educational_Qualification','Hi
aerofit_df['Education_Groups'] = pd.cut(aerofit_df['Education'],bins_income,labels = labels
```

In [199]:

```
# Concat Gender and Marital Status
aerofit_df['Gender_Marital_Status_Age'] = aerofit_df['Gender'].astype(str) + '-' + aerofit_
```

In [202]:

```
aerofit_df['Gender_Marital_Status_Age'].unique()
```

Out[202]:

```
array(['Male-Single-Teens', 'Female-Partnered-Teens',
       'Male-Partnered-Teens', 'Female-Partnered-20s', 'Male-Single-20s',
       'Female-Single-20s', 'Male-Partnered-20s', 'Male-Partnered-30s',
       'Female-Single-30s', 'Female-Partnered-30s', 'Male-Single-30s',
       'Male-Partnered-Above 40s', 'Female-Single-Above 40s',
       'Female-Partnered-Above 40s', 'Male-Single-Above 40s'],
      dtype=object)
```

In [205]:

```
aerofit_df.head()
```

Out[205]:

›me_Groups	Gender_Marital_Status	Education_Groups	Income_Groups_Code	Educa
Low Income Group	Male-Single	Average_Educational_Qualification	Income:LIG	
Lower-Middle ıcome Group	Male-Single	Average_Educational_Qualification	Income:LMIG	
Lower-Middle ıcome Group	Female-Partnered	Average_Educational_Qualification	Income:LMIG	
Lower-Middle ıcome Group	Male-Single	Low_Educational_Qualification	Income:LMIG	
Lower-Middle ıcome Group	Male-Partnered	Low_Educational_Qualification	Income:LMIG	

In [203]:

```
aerofit_df['Income_Groups_Code'] = aerofit_df['Income_Groups'].map({'Low Income Group': 'Ir
                                                                    'Lower-Middle Income Gr
                                                                    'Upper-Middle Income Gr
                                                                     'High Income Group': '
```

In [204]:

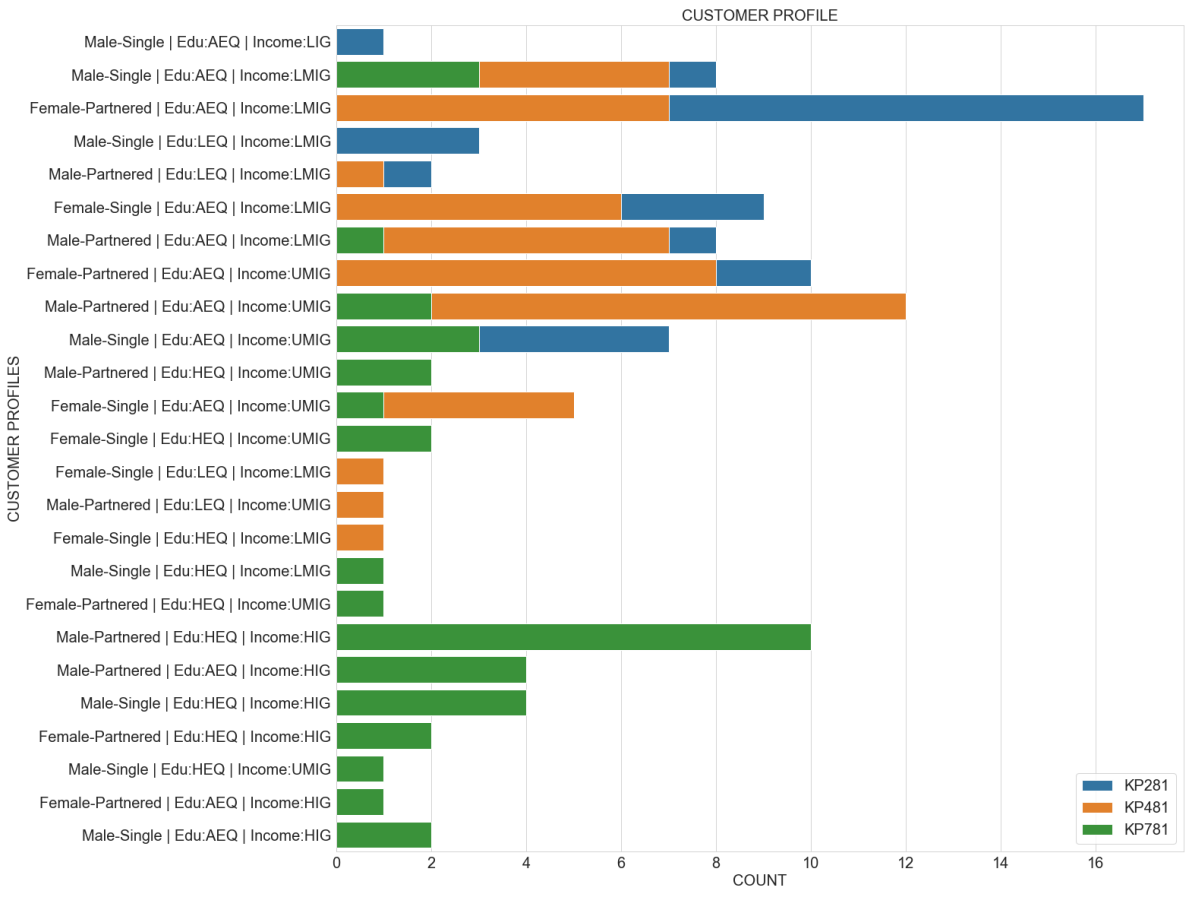
```
aerofit_df['Education_Groups_code'] = aerofit_df['Education_Groups'].map({'Low_Educational_
                                                                          'Average_Educational_Qu
                                                                          'High_Educational_Quali
```

In [233]:

```
aerofit_df['Customer_profile'] = aerofit_df['Gender_Marital_Status'].astype(str) + ' | ' +
```

In [273]:

```
plt.figure(figsize=(20,20))
sns.countplot(y = aerofit_df["Customer_profile"], hue = aerofit_df['Product'], dodge=False)
plt.legend(loc="lower right", fontsize = 20)
plt.yticks( fontsize = 20)
plt.xticks( fontsize = 20)
plt.ylabel('CUSTOMER PROFILES',  fontsize = 20)
plt.xlabel('COUNT',  fontsize = 20)
plt.title('CUSTOMER PROFILE', fontsize = 20)

plt.show()
```



Calculation of Marginal Probabilities, Joint Probabilities, Conditional

Probabilities

In [276]:

```
#Income_groups
pd.crosstab(index=aerofit_df['Product'], columns=[aerofit_df['Income_Groups']],margins=True
```

Out[276]:

Income_Groups	Low Income Group	Lower-Middle Income Group	Upper-Middle Income Group	High Income Group	All
Product					
KP281	1	47	32	0	80
KP481	0	30	30	0	60
KP781	0	5	12	23	40
All	1	82	74	23	180

Probabilistic Insights

In [283]:

```
#Probabilty of purchasing a product by differnt Income Groups: (Marginal Probability)
print("Probabilty of Lower-Middle Income Group purchasing a product : {}" .format(round(82/
print("Probabilty of Upper-Middle Income Group purchasing a product : {}" .format(round(74/
print("Probabilty of High Income Group purchasing a product : {}" .format(round(23/180,2)))
```

```
Probabilty of Lower-Middle Income Group purchasing a product : 0.46
Probabilty of Upper-Middle Income Group purchasing a product : 0.41
Probabilty of High Income Group purchasing a product : 0.13
```

In [288]:

```
#Probabilty of purchasing KP781 & KP481 product by differnt Income Groups: (Joint Probabili
print("Probabilty of Lower-Middle Income Group purchasing a product : {}" .format(round(rou
print("Probabilty of Upper-Middle Income Group purchasing a product : {}" .format(round(rou
print("Probabilty of High Income Group purchasing a product : {}" .format(round(23/23,2)*ro
```

```
Probabilty of Lower-Middle Income Group purchasing a product : 0.01
Probabilty of Upper-Middle Income Group purchasing a product : 0.01
Probabilty of High Income Group purchasing a product : 0.0
```

In [289]:

```
#Product and Gender
pd.crosstab(index=aerofit_df['Product'], columns=[aerofit_df['Gender']],margins=True)
```

Out[289]:

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

Probabilty of Female customer buying treadmill given that Product is KP781

- P(A|B) = P(A,B)/P(B)
- P(Female|KP781) = P(Female,KP781)/P(KP781)

In [291]:

```
print("Probabilty of Female customer buying treadmill given that Product is KP781 : {}" .fo
```

```
Probabilty of Female customer buying treadmill given that Product is KP781 :
0.17
```

In [292]:

```
print("Probabilty of Male customer buying treadmill given that Product is KP781 : {}" .form
```

```
Probabilty of Male customer buying treadmill given that Product is KP781 :
0.82
```

Recommendations:

- KP281 & kp481 are popular with customers earning USD below 60,000 and can be offered by these companies as affordable models.
- KP781 should be marketed as a Premium Model and marketing it to high income groups and educational over 20 years market segments could result in more sales.
- Aerofit should conduct market research to determine if it can diversify it's product accross all the customer segemnts.
- The KP781 is a premium model, so it is ideally suited for sporty people who are married and having a higher fitness

In [ ]: