About 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.

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.

Perform descriptive analytics to create a customer profile for each AeroFit treadmill product by developing appropriate tables and charts. For each AeroFit treadmill product, construct two-way contingency tables and compute all conditional and marginal probabilities along with their insights/impact on the business.

Dataset

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.

Analysing basic metrics

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

In [ ]:
```
!gdown 1fZMfXRvPISsZjrZod73RWsRVlG38ZnvO
```

```
Downloading...
From: https://drive.google.com/uc?id=1fZMfXRvPISsZjrZod73RWsRVlG38ZnvO
To: /content/aerofit_treadmill.csv
100% 7.28k/7.28k [00:00<00:00, 31.2MB/s]
```

In [ ]:
```
data = pd.read_csv("aerofit_treadmill.csv")
```

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

Out[ ]:

	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 [ ]:
```
data.shape
```

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

In [ ]:
```
data.columns.to_list()
```

Out[ ]:
```
['Product',
 'Age',
 'Gender',
 'Education',
 'MaritalStatus',
 'Usage',
 'Fitness',
 'Income',
 'Miles']
```

In [ ]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180 entries, 0 to 179
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   Product        180 non-null    object
 1   Age            180 non-null    int64 
 2   Gender         180 non-null    object
 3   Education      180 non-null    int64 
 4   MaritalStatus  180 non-null    object
 5   Usage          180 non-null    int64 
 6   Fitness        180 non-null    int64 
 7   Income         180 non-null    int64 
 8   Miles          180 non-null    int64 
dtypes: int64(6), object(3)
memory usage: 12.8+ KB
```

In [ ]:
```
(data.isnull()).sum()
```

Out[ ]:

	0

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

dtype: int64

In [ ]:
```
np.median(data["Income"])
```

Out[ ]: 50596.5

In [ ]:
```
data.describe()
```

Out[ ]:

	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

There are 180 rows and 9 columns in the Aerofit dataset.

- The mean age is around 28 years.
- The maximum age is 50 years.
- The Highest Educational qualification is 21.
- Maximum customers are using 3 in a week.
- The Mean fitness of the customers is 3.
- The Mean Income is about 53719 and The 50% of income is around 50,596.
- The Income having some outlier data.
- The average number of miles the customer expects to walk/run each week 94 Miles.

Binning into category

Age Column Categorizing the values in age column in 4 different buckets:

- Young Adult: from 18 - 25
- Adults: from 26 - 35
- Middle Aged Adults: 36-45
- Elder :46 and above

Education Column Categorizing the values in education column in 3 different buckets:

- Primary Education: upto 12
- Secondary Education: 13 to 15
- Higher Education: 16 and above

Income Column Categorizing the values in Income column in 4 different buckets:

- Low Income - Upto 40,000
- Moderate Income - 40,000 to 60,000
- High Income - 60,000 to 80,000
- Very High Income - Above 80,000

Miles column

- Light Activity - Upto 50 miles
- Moderate Activity - 51 to 100 miles
- Active Lifestyle - 101 to 200 miles
- Fitness Enthusiast - Above 200 miles

In [ ]:
```
# Categorizing of Age column
age_range = [17, 25, 35, 45, float("inf") ]
age_bins = ["Young_Adult", "Adults", "Middle_Aged_Adults", "Elder"]
data["Age_cat"] = pd.cut(data["Age"], bins=age_range, labels=age_bins, ordered=True)

# Categorizing of Education column
edu_range = [0, 12, 15, float("inf")]
edu_bins = ["primary", "secondary", "higher"]
data["Education_cat"] = pd.cut(data["Education"], bins=edu_range, labels=edu_bins, ordered=True)

# Categorizing of Income column
income_range = [0, 40000, 60000, 80000, float("inf")]
income_bins = ["low", "Moderate", "High","very_high"]
data["income_cat"] = pd.cut(data["Income"], bins=income_range, labels=income_bins, ordered=True)

# Categorizing of Miles column
miles_range = [0, 50, 100, 200,float("inf")]
miles_bins = ["Light_Activity", "Moderate_Activity", "High_Activity", "Fitness Enthusiast"]
data["miles_cat"] = pd.cut(data["Miles"], bins=miles_range, labels=miles_bins, ordered=True)
```

In [ ]:
```
data["fitness_cat"] = data['Fitness']
data["fitness_cat"].replace({1:"Poor Shape", 2:"Bad Shape", 3:"Average Shape", 4:"Good Shape", 5:"Excellent Shape"},
```

```
<ipython-input-12-ef2f430f9c47>:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series thr
ough chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which
we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or
df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  data["fitness_cat"].replace({1:"Poor Shape", 2:"Bad Shape", 3:"Average Shape", 4:"Good Shape", 5:"Excellent Shap
e"},inplace=True)
```

In [ ]:
```
data["fitness_cat"].value_counts()
```

Out[ ]:

fitness_cat	count
Average Shape	97
Excellent Shape	31
Bad Shape	26
Good Shape	24
Poor Shape	2

dtype: int64

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

Out[ ]:

miles_cat	proportion
Moderate_Activity	0.538889

High_Activity	0.333333
Light_Activity	0.094444
Fitness Enthusiast	0.033333

dtype: float64

In [ ]:
```
data["Education_cat"].value_counts()
```

Out[ ]:

Education_cat	count
higher	112
secondary	65
primary	3

dtype: int64

In [ ]:
```
data["income_cat"].value_counts()
```

Out[ ]:

income_cat	count
Moderate	106
low	32
High	23
very_high	19

dtype: int64

Non-Graphical Analysis

In [ ]:
```
data["Product"].unique().tolist()
```

Out[ ]:
```
['KP281', 'KP481', 'KP781']
```

There are Three Unique products. They are KP281, KP481, KP781

In [ ]:
```
data["Product"].value_counts()
```

Out[ ]:

Product	count
KP281	80
KP481	60
KP781	40

dtype: int64

In [ ]:
```
data["Gender"].unique().tolist()
```

Out[ ]:
```
['Male', 'Female']
```

In [ ]:
```
data["Gender"].value_counts()
```

Out[ ]:

Gender	count

Male	104
Female	76

dtype: int64

In [ ]:
```
data["MaritalStatus"].unique().tolist()
```

Out[ ]:
```
['Single', 'Partnered']
```

In [ ]:
```
data["MaritalStatus"].value_counts()
```

Out[ ]:

MaritalStatus	count
Partnered	107
Single	73

dtype: int64

In [ ]:
```
data["MaritalStatus"].value_counts(normalize=True).map(lambda x: round(x*100,2)).reset_index()
```

Out[ ]:

	MaritalStatus	proportion
0	Partnered	59.44
1	Single	40.56

The Martial Status consisiting of two unique values

1. Partnered
2. Single

The Partnered/Married are covering 59.44% and Single are 40.56%

In [ ]:
```
data["Usage"].value_counts(normalize=True).map(lambda x: x*100).reset_index().rename({"Usage": "Days_for_week"}, ax
```

Out[ ]:

	Days_for_week	proportion
0	3	38.333333
1	4	28.888889
2	2	18.333333
3	5	9.444444
4	6	3.888889
5	7	1.111111

- Around customers are using 3 days are 39%
- Around customers are using 4 days per week are 29%

In [ ]:
```
data.groupby("MaritalStatus")["Product"].count()
```

Out[ ]:

MaritalStatus	Product
Partnered	107
Single	73

dtype: int64

In [ ]:
```
data.groupby("Age_cat")["Product"].count()
```

```
<ipython-input-26-ad917a2954bd>:1: FutureWarning: The default of observed=False is deprecated and will be changed to
True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the fut
ure default and silence this warning.
  data.groupby("Age_cat")["Product"].count()
```

Out[ ]:

Age_cat	Product
Young_Adult	79
Adults	73
Middle_Aged_Adults	22
Elder	6

dtype: int64

In [ ]:
```
data.groupby("Education_cat")["Product"].count()
```

```
<ipython-input-27-7a638f69306f>:1: FutureWarning: The default of observed=False is deprecated and will be changed to
True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the fut
ure default and silence this warning.
  data.groupby("Education_cat")["Product"].count()
```

Out[ ]:

Education_cat	Product
primary	3
secondary	65
higher	112

dtype: int64

In [ ]:
```
data.groupby("income_cat")["Product"].count()
```

```
<ipython-input-28-3fd505b8a8d7>:1: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  data.groupby("income_cat")["Product"].count()
```

Out[ ]:

income_cat	Product
low	32
Moderate	106
High	23
very_high	19

dtype: int64

In [ ]:
```
data.groupby("fitness_cat")["Product"].count()
```

Out[ ]:

fitness_cat	Product
Average Shape	97
Bad Shape	26
Excellent Shape	31
Good Shape	24
Poor Shape	2

dtype: int64

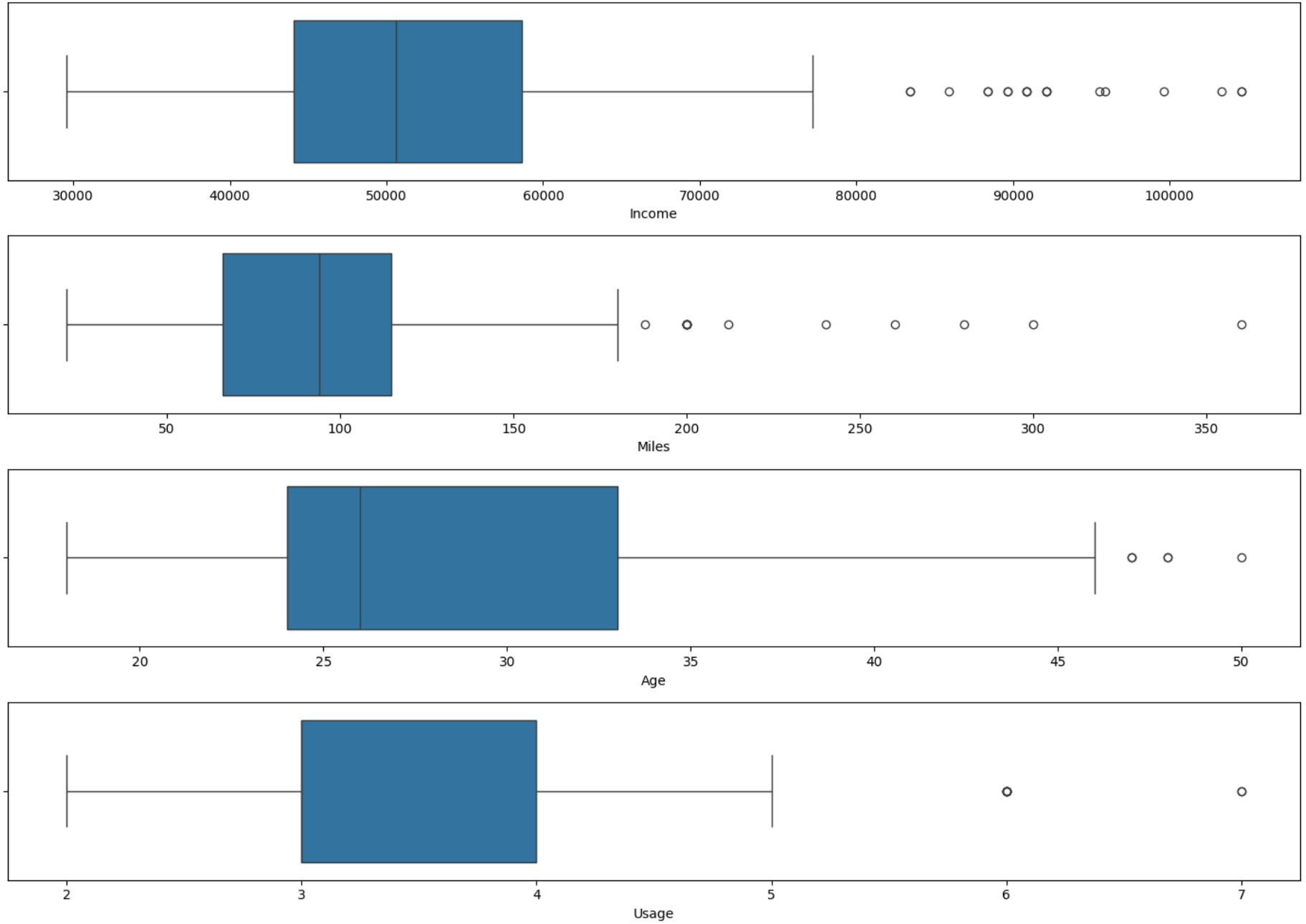
In [ ]:
```
data.describe()
```

Out[ ]:

	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

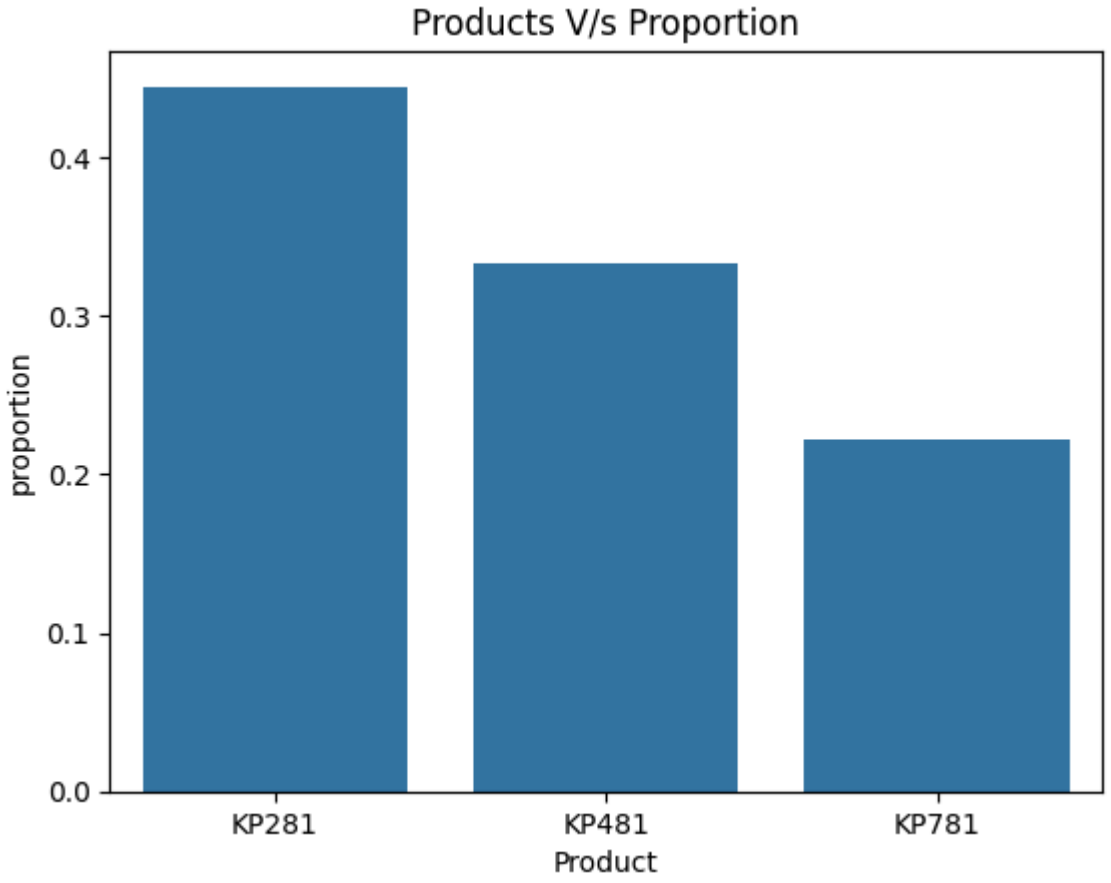
In [ ]:
```
plt.figure(figsize=(14, 10))


plt.subplot(4, 2, (1,2))
sns.boxplot(data=data, x= data["Income"])


plt.subplot(4, 2, (3,4))
sns.boxplot(data=data, x= data["Miles"])

plt.subplot(4, 2, (5,6))
sns.boxplot(data=data, x= data["Age"])

plt.subplot(4, 2, (7,8))
sns.boxplot(data=data, x= data["Usage"])
```

```
plt.tight_layout()
plt.show()
```

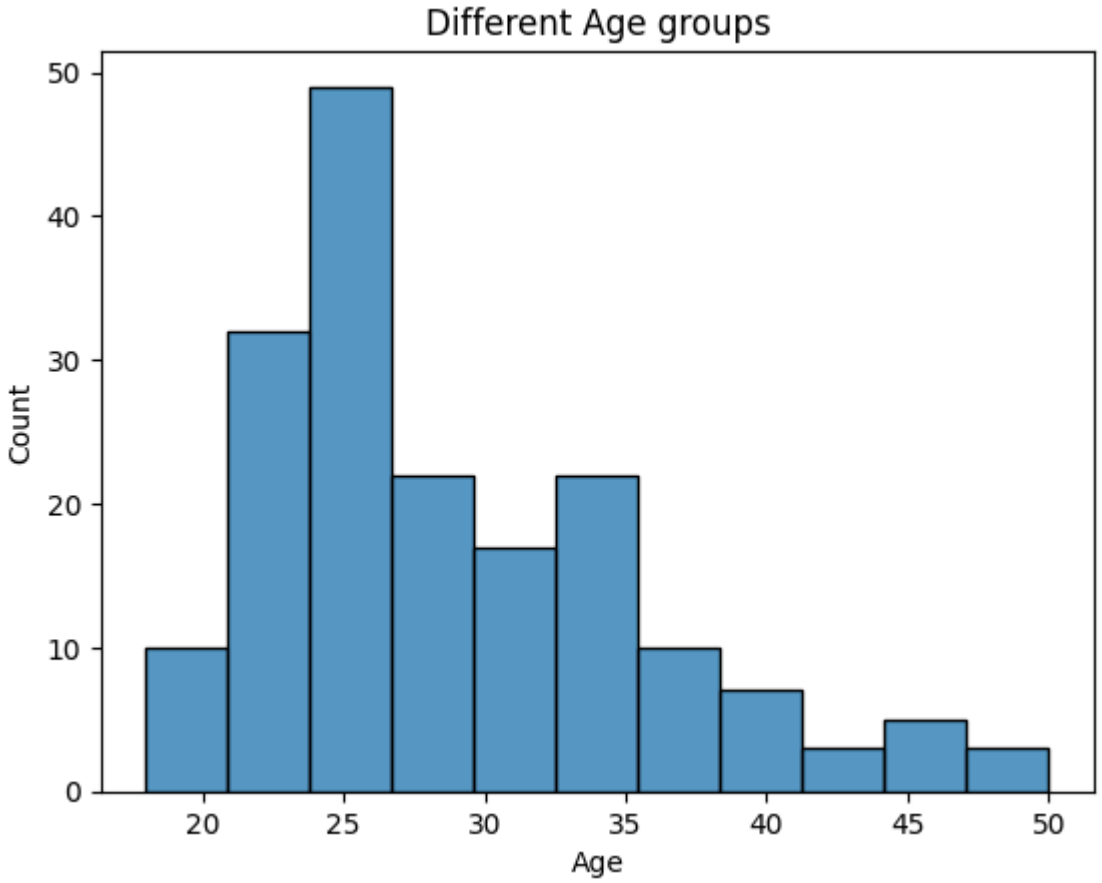

30000
40000
50000
60000
70000
80000
90000
100000
Income
50
100
150
200
250
300
350
Miles
20
25
30
35
40
45
50
Age
2
3
4
5
6
7
Usage

Visual Analysis - Univariate & Bivariate

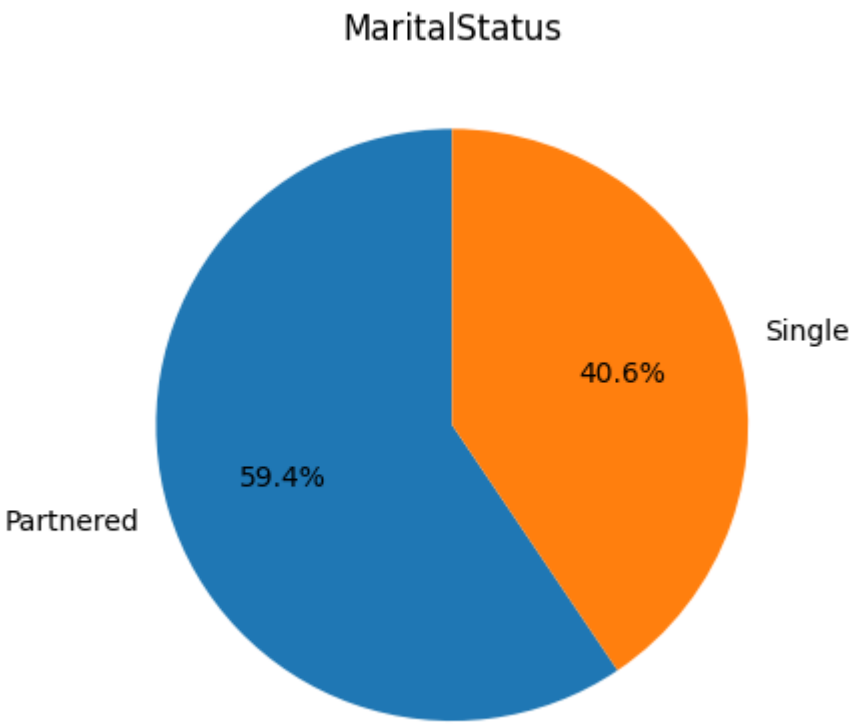
In [ ]:
```
sns.countplot(data = data, x=data["Product"], stat='proportion')
plt.title("Products V/s Proportion")
plt.show()
```



In [ ]:
```
sns.histplot(data = data, x=data["Age"])
plt.title("Different Age groups")
plt.show()
```

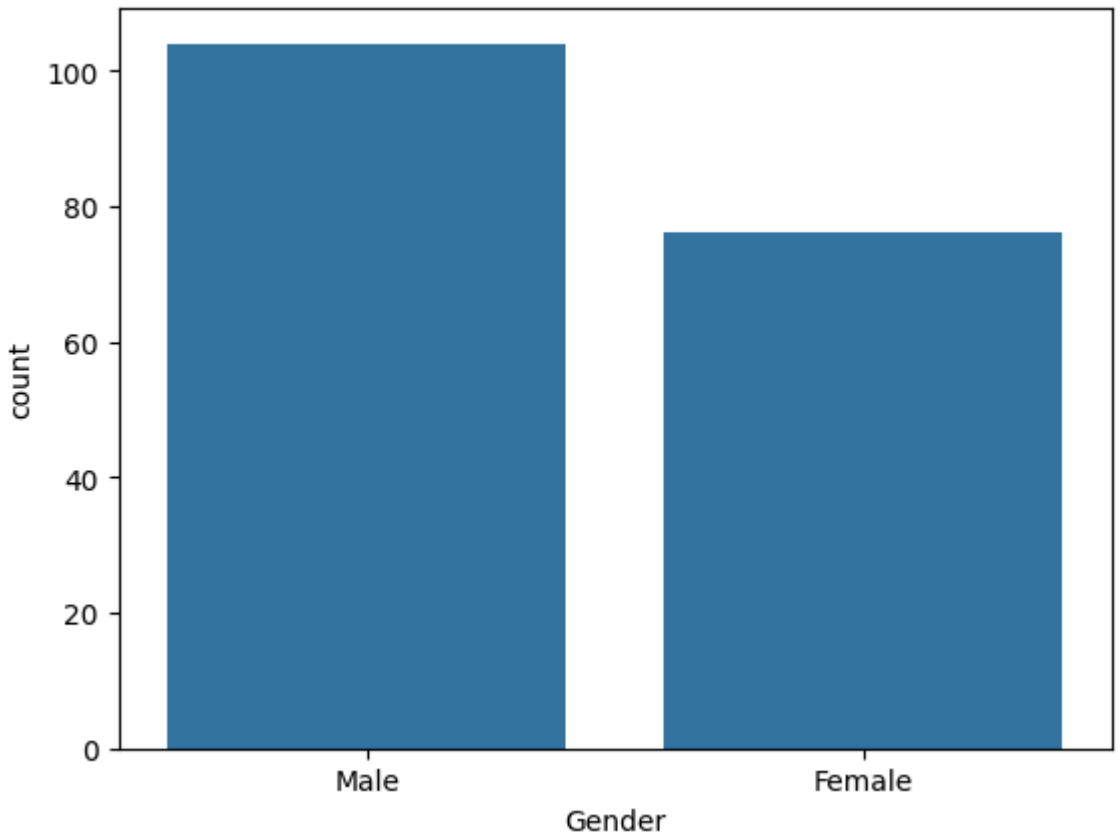


In [ ]:
```
label = data["MaritalStatus"].value_counts().index
values = data["MaritalStatus"].value_counts().values

plt.pie(values, labels=label, autopct="%1.1f%%",startangle=90)
plt.title("MaritalStatus")
plt.show()
```

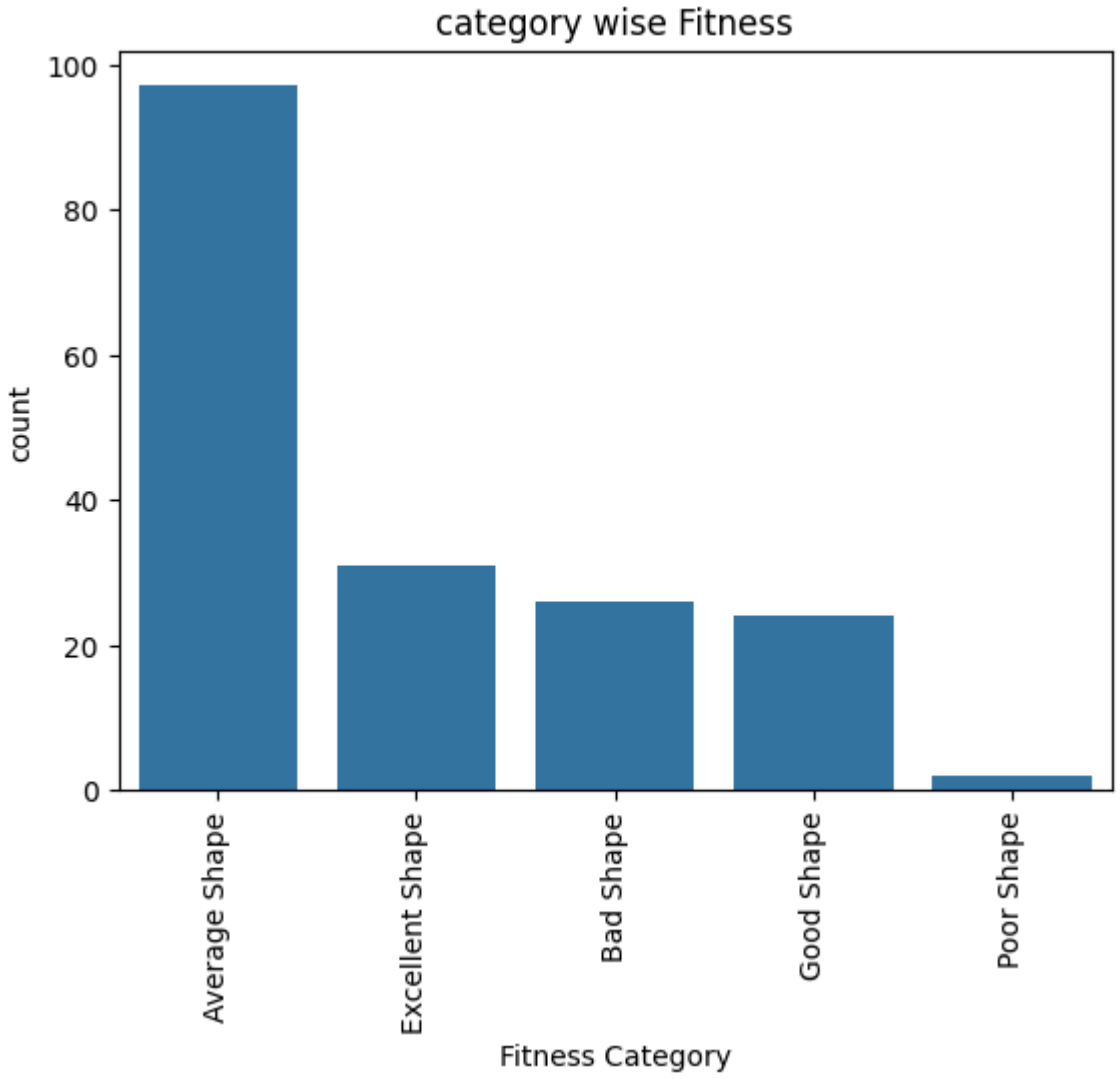


In [ ]:
```
sns.countplot(data=data, x=data["Gender"])
```

Out[ ]:
```
<Axes: xlabel='Gender', ylabel='count'>
```

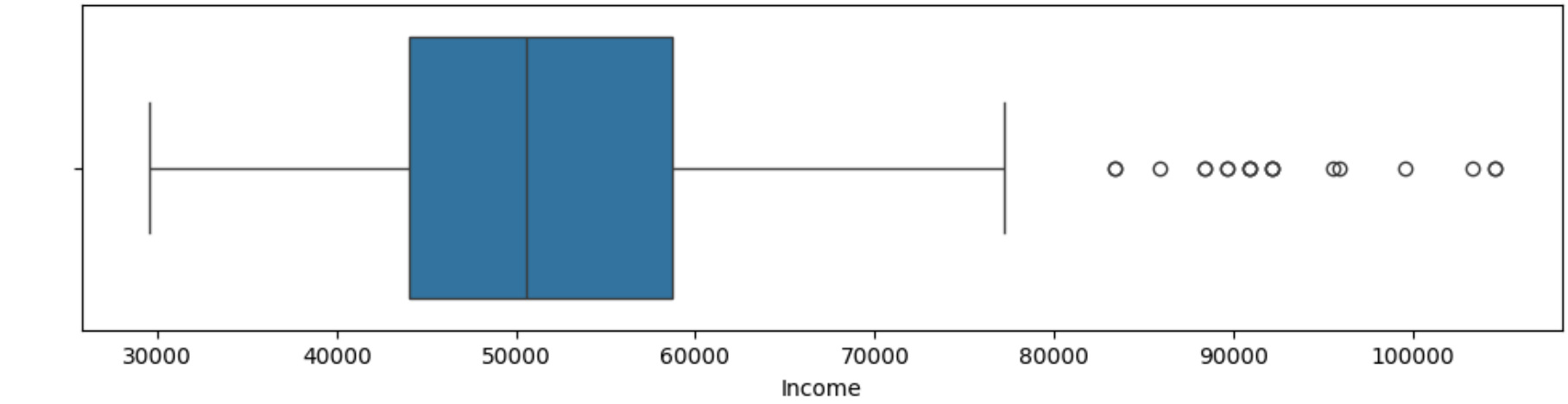


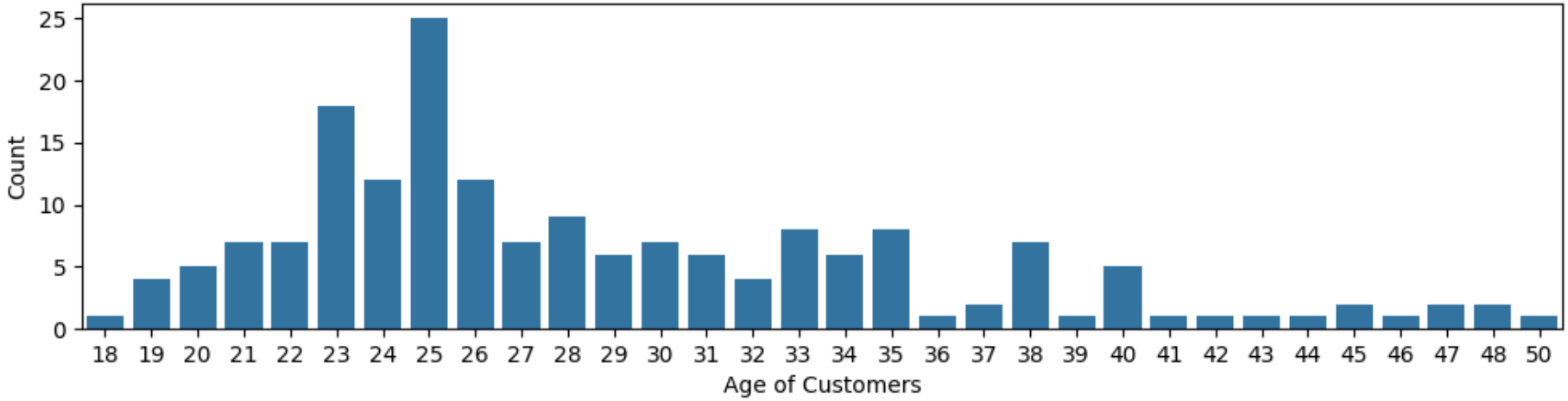
In [ ]:
```
order = data["fitness_cat"].value_counts(ascending=False).index
sns.countplot(data= data, x=data["fitness_cat"], order=order,fill=True)
plt.xlabel("Fitness Category")
plt.ylabel("count")
plt.xticks(rotation=90)
plt.title("category wise Fitness")
plt.show()
```

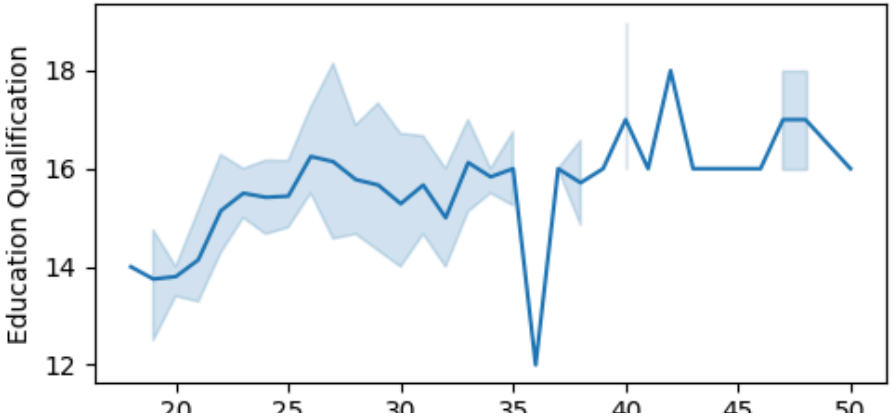


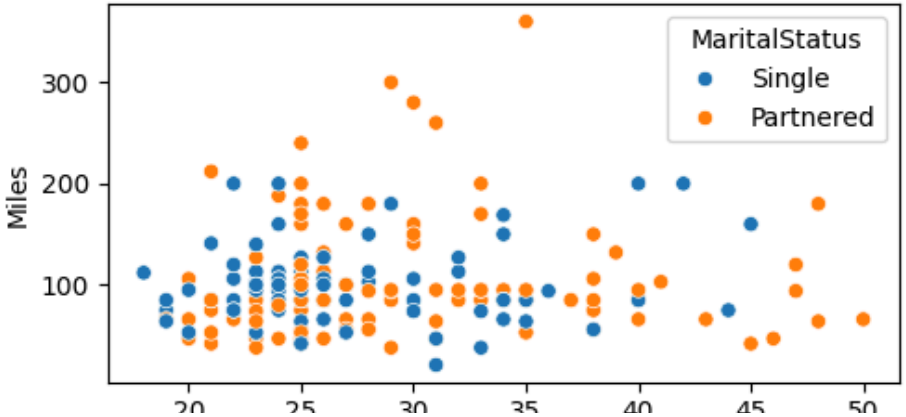
In [ ]:
```
plt.figure(figsize=(10,8))

plt.subplot(3,2, (1, 2))
sns.boxplot(data=data, x=data["Income"], fill=True, orient="V")

plt.subplot(3,2, (3,4))
sns.countplot(data=data, x = data["Age"])
```

```
plt.xlabel("Age of Customers")
plt.ylabel("Count")

plt.subplot(3,2, 5)
sns.lineplot(data=data, x=data["Age"] ,y = data["Education"], estimator='mean', legend='auto')
plt.xlabel("Age of Customers")
plt.ylabel("Education Qualification")


plt.subplot(3,2, 6)
sns.scatterplot(data=data, y=data["Miles"], x=data["Age"], hue=data["MaritalStatus"])
plt.xlabel("Age of Customers")
plt.ylabel("Miles")



plt.tight_layout()
plt.show()
```


30000
40000
50000
60000
70000
80000
90000
100000
Income

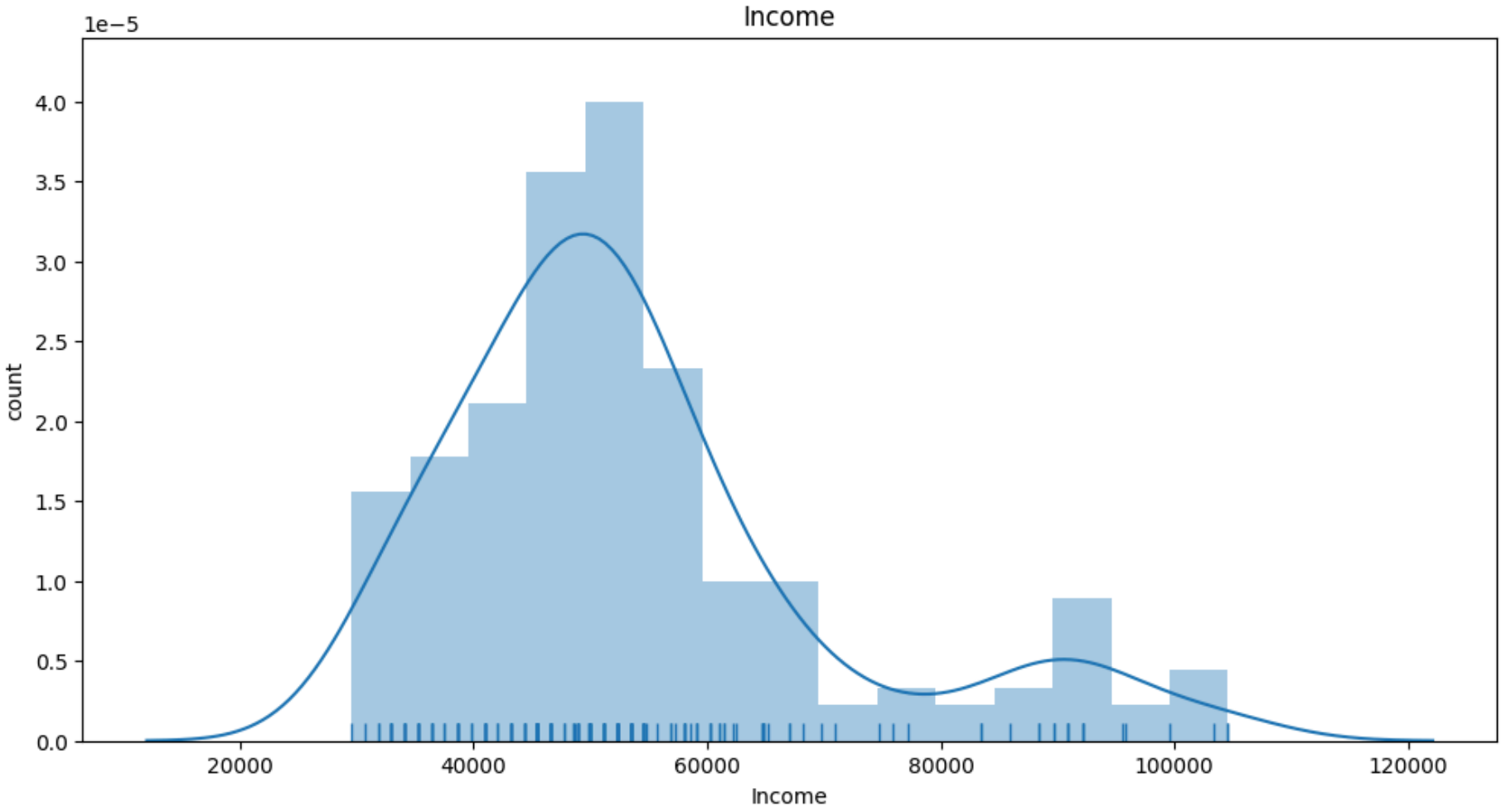

Count
0
5
10
15
20
25
18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 50
Age of Customers


Education Qualification
12
14
16
18
20
25
30
35
40
45
50

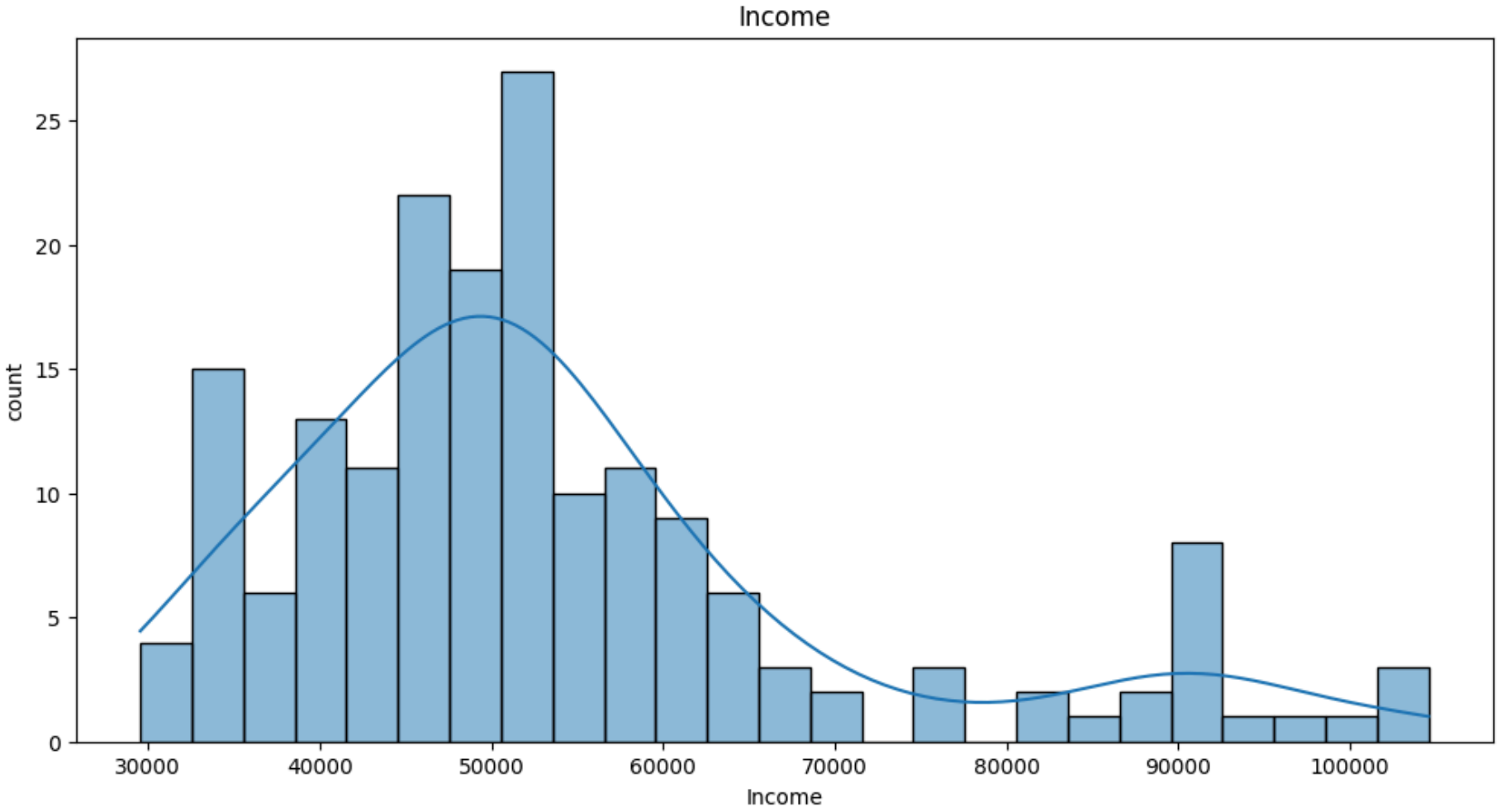

Miles
100
200
300
MaritalStatus
Single
Partnered
20
25
30
35
40
45
50

In [ ]:
```
plt.figure(figsize=(12,6))
sns.distplot(data["Income"],rug=True)
plt.xlabel("Income")
plt.ylabel("count")
plt.title("Income")
plt.show()
```

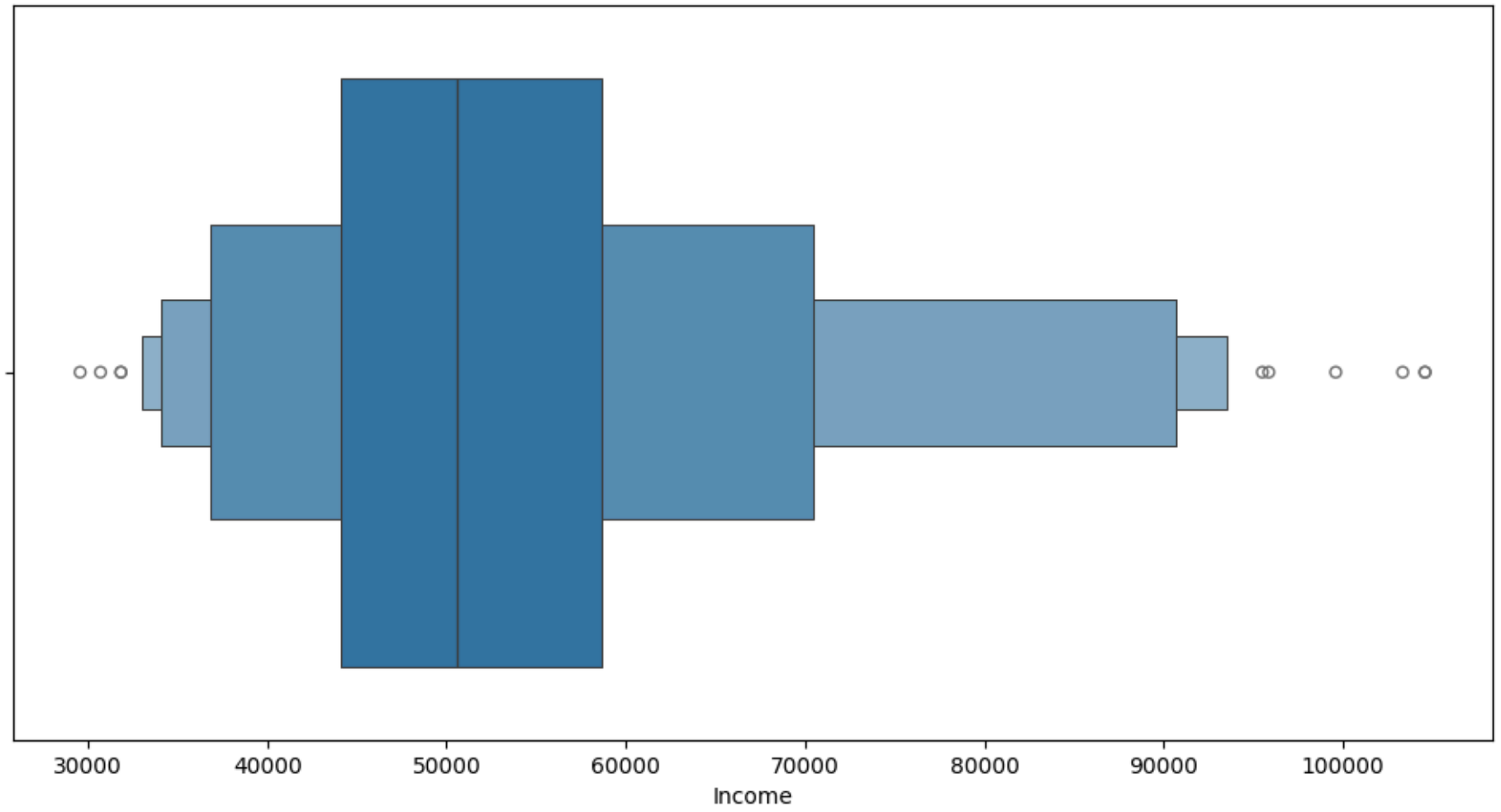
```
<ipython-input-38-dfe1b92731da>:2: UserWarning:

`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(data["Income"],rug=True)
```



In [ ]:
```
plt.figure(figsize=(12,6))
sns.histplot(data=data, x= data["Income"], bins=25, kde=True)
plt.xlabel("Income")
plt.ylabel("count")
plt.title("Income")
plt.show()
```



In [ ]:
```
plt.figure(figsize=(12,6))
sns.boxenplot(data=data, x=data.Income)
plt.show()
```



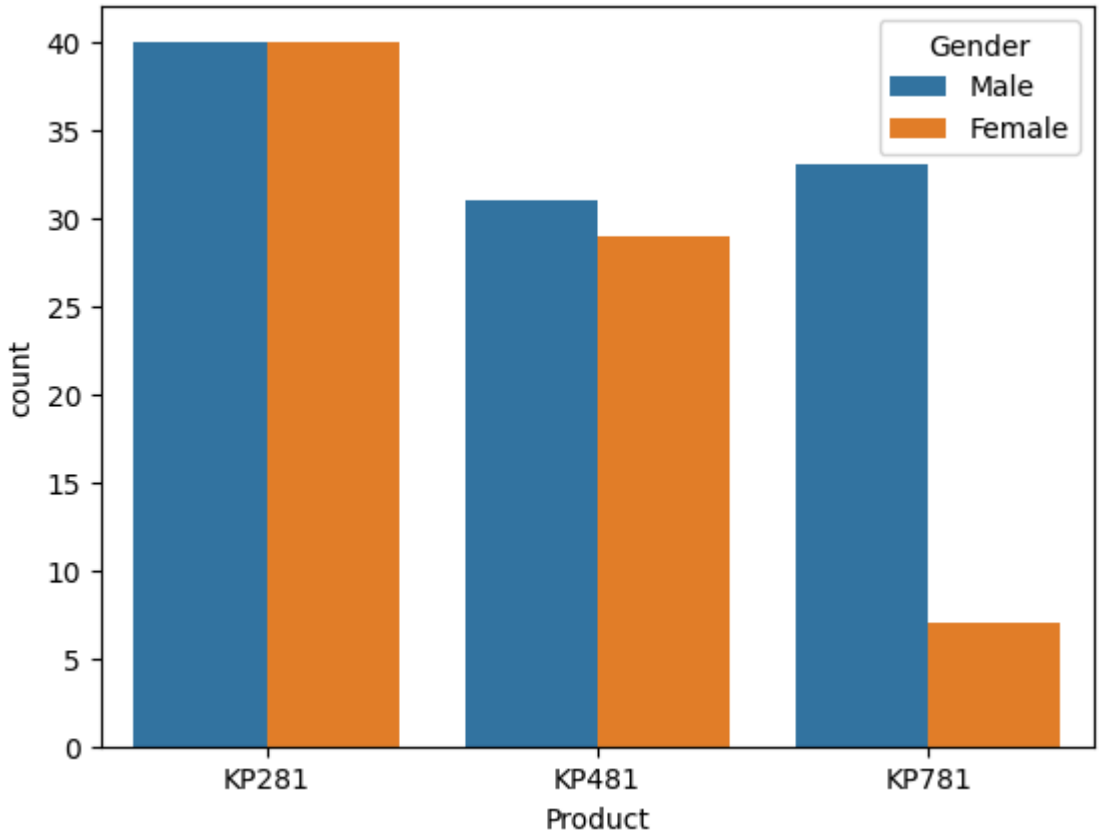
In [ ]:
```
data.head(1)
```

Out[ ]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles	Age_cat	Education_cat	income_cat	miles_cat	fitn
0	KP281	18	Male	14	Single	3	4	29562	112	Young_Adult	secondary	low	High_Activity	

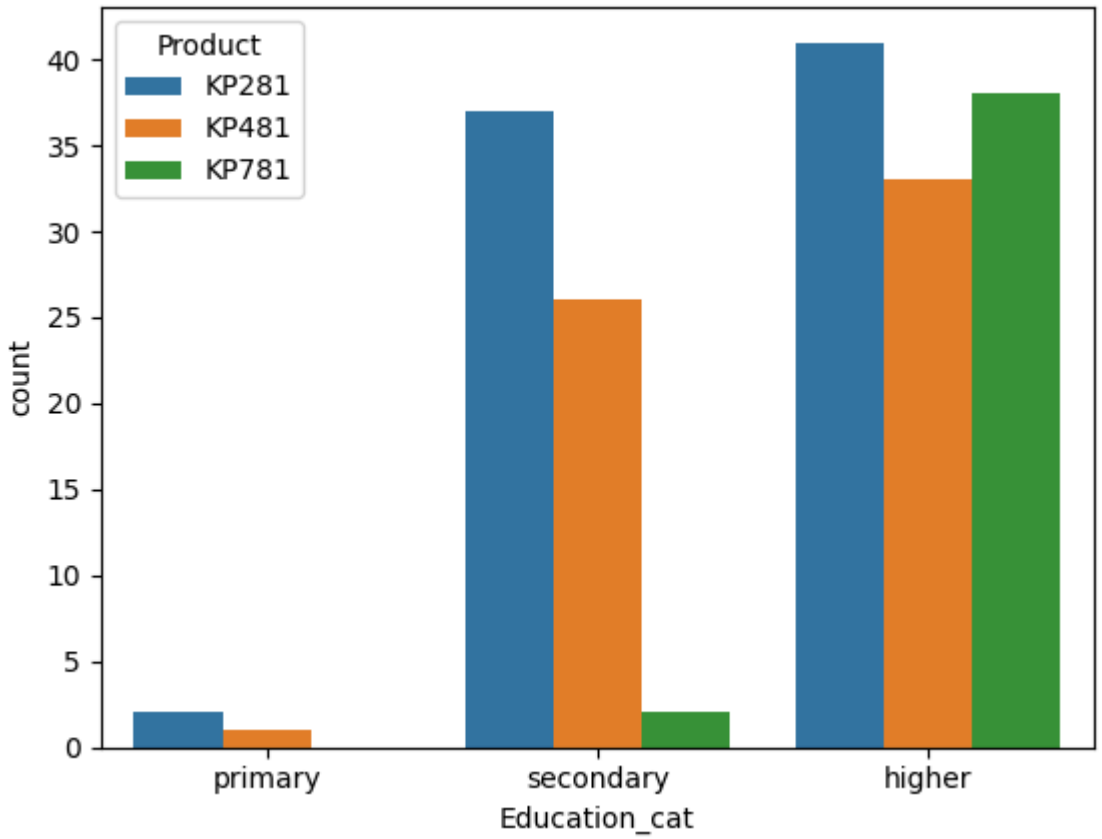
In [ ]:
```
sns.countplot(data=data, x=data["Product"], hue= data.Gender, dodge=True, stat='count')
```

Out[ ]:
```
<Axes: xlabel='Product', ylabel='count'>
```



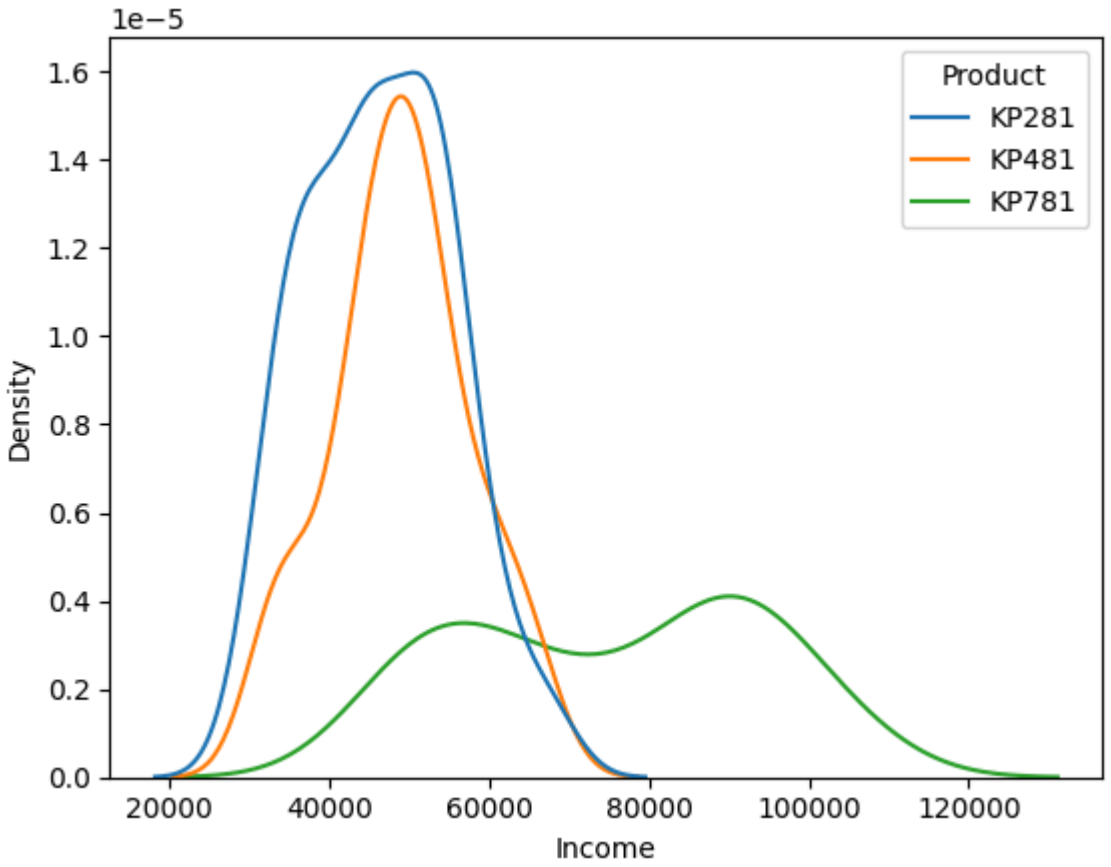
In [ ]:
```
sns.countplot(data=data, x=data["Education_cat"], hue= data.Product, dodge=True, stat='count', legend="auto")
```

Out[ ]:
```
<Axes: xlabel='Education_cat', ylabel='count'>
```



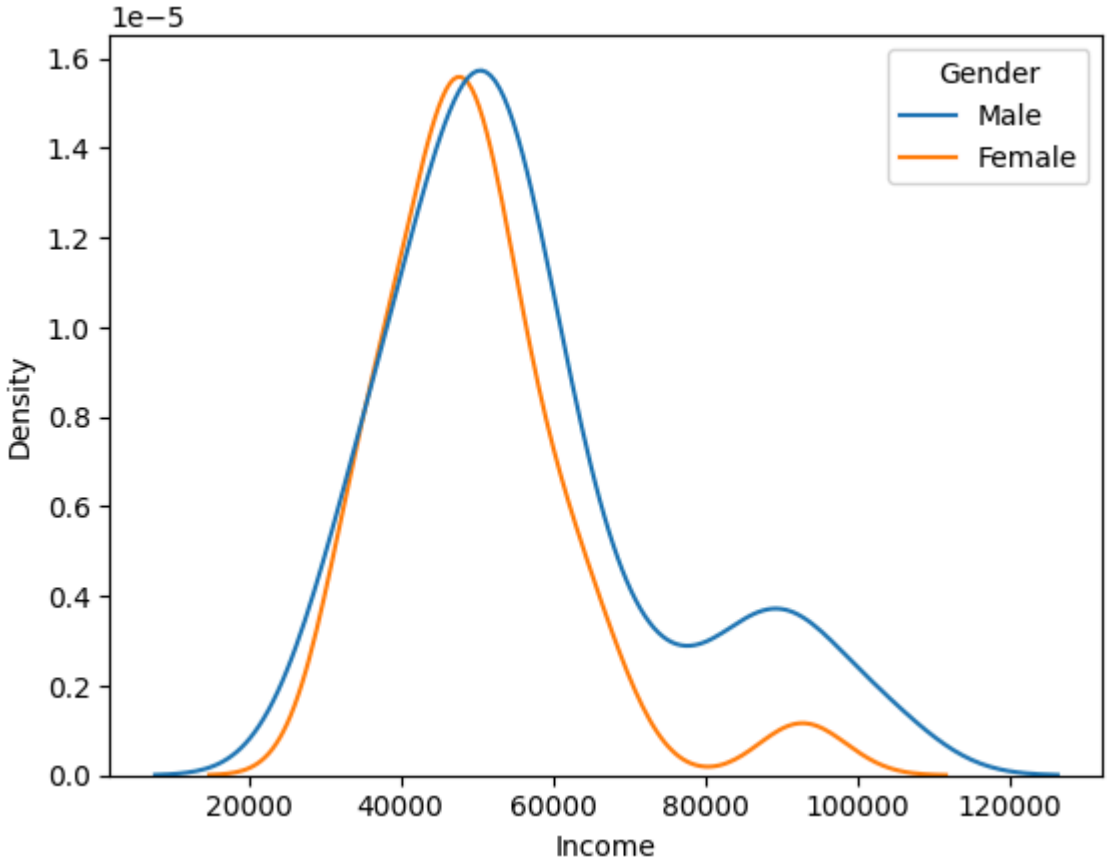
In [ ]:
```
sns.kdeplot(data=data, x= data.Income, hue=data.Product)
```

Out[ ]:
```
<Axes: xlabel='Income', ylabel='Density'>
```



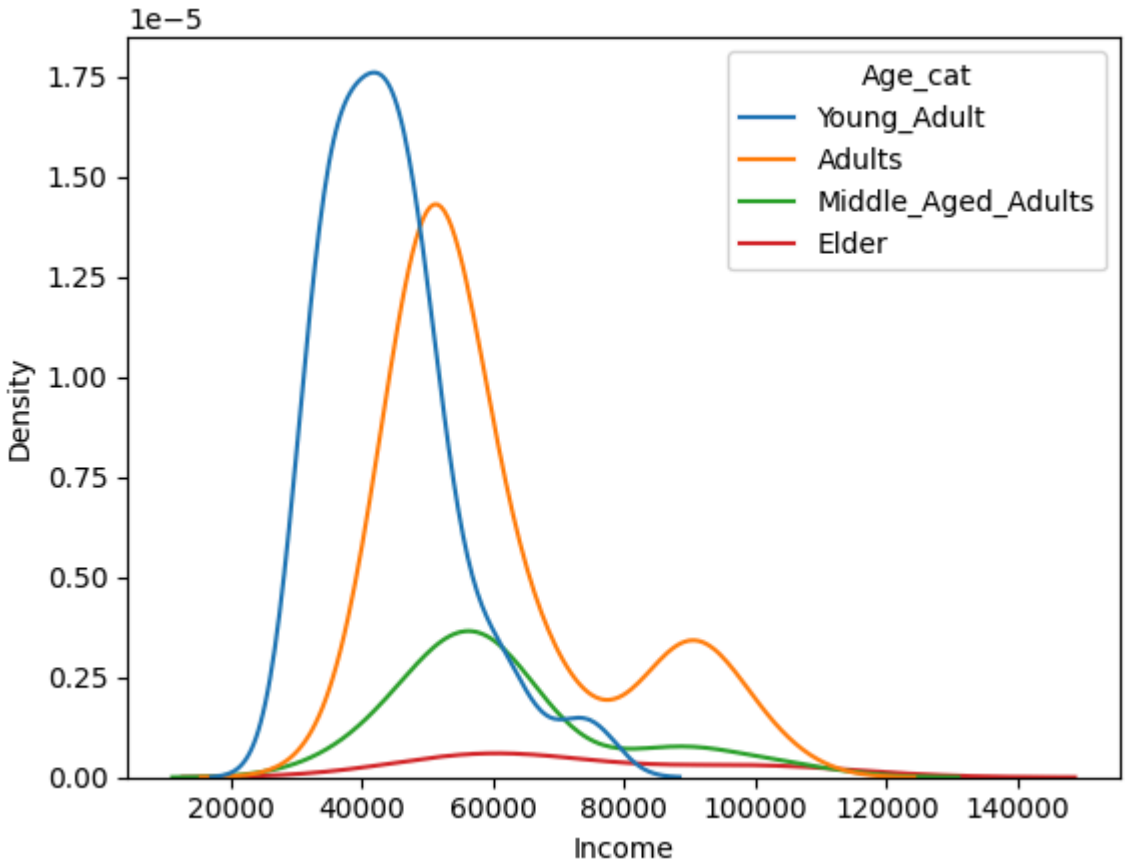
In [ ]:
```
sns.kdeplot(data=data, x= data.Income, hue=data.Gender)
```

Out[ ]:
```
<Axes: xlabel='Income', ylabel='Density'>
```



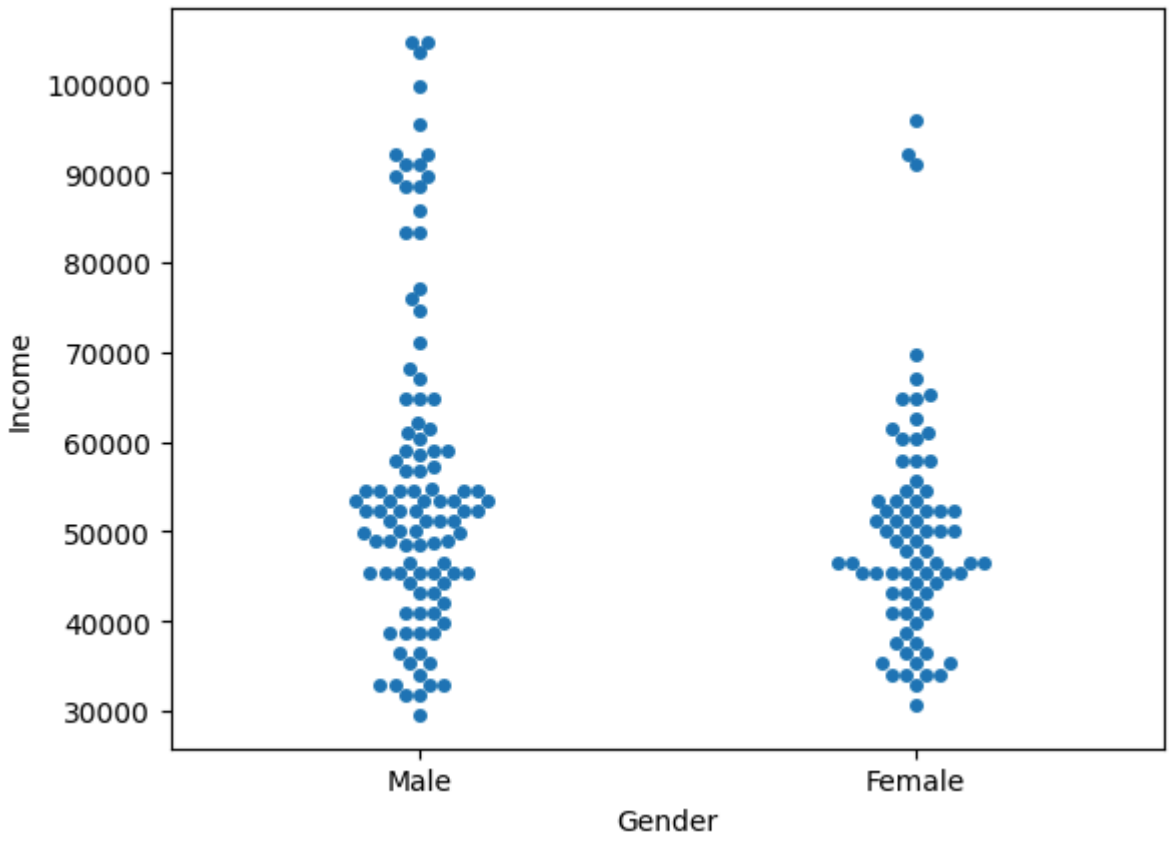
In [ ]:
```
sns.kdeplot(data=data, x= data.Income, hue=data.Age_cat)
```

Out[ ]:
```
<Axes: xlabel='Income', ylabel='Density'>
```



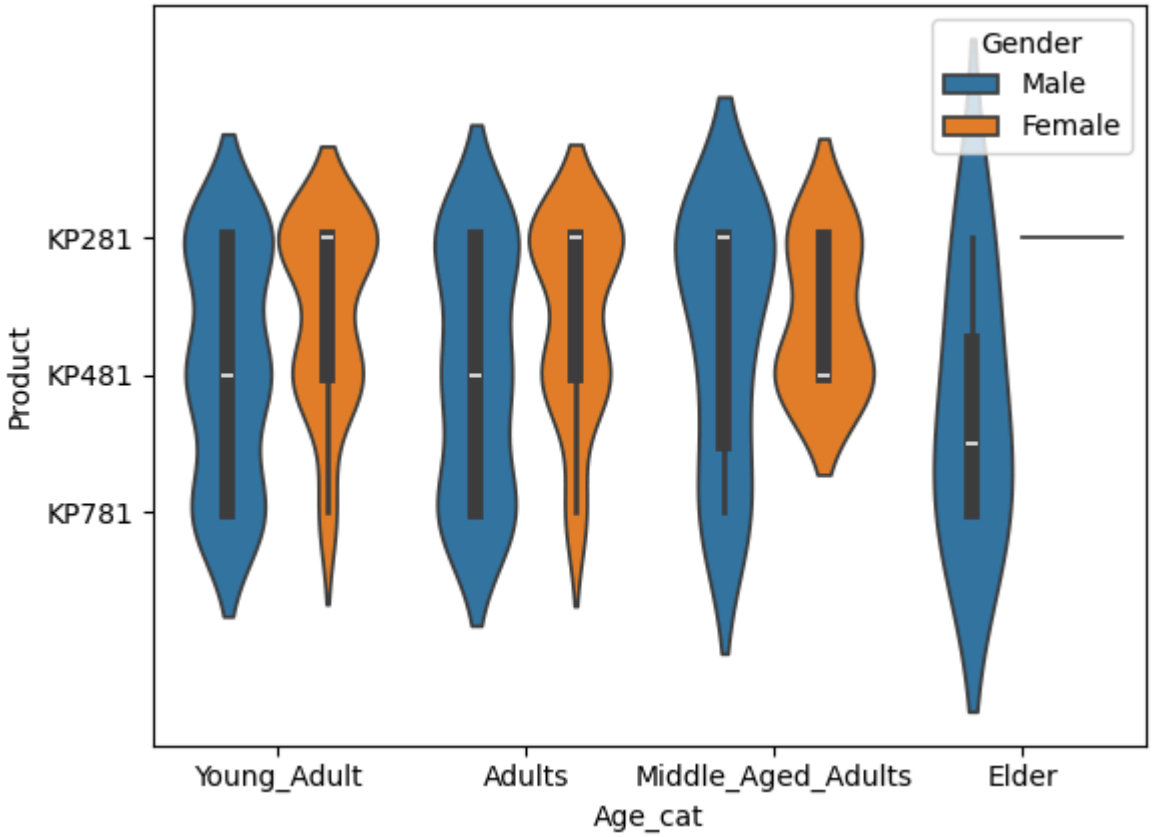
In [ ]:
```
sns.swarmplot(data= data, x=data.Gender, y=data.Income)
```

Out[ ]:
```
<Axes: xlabel='Gender', ylabel='Income'>
```



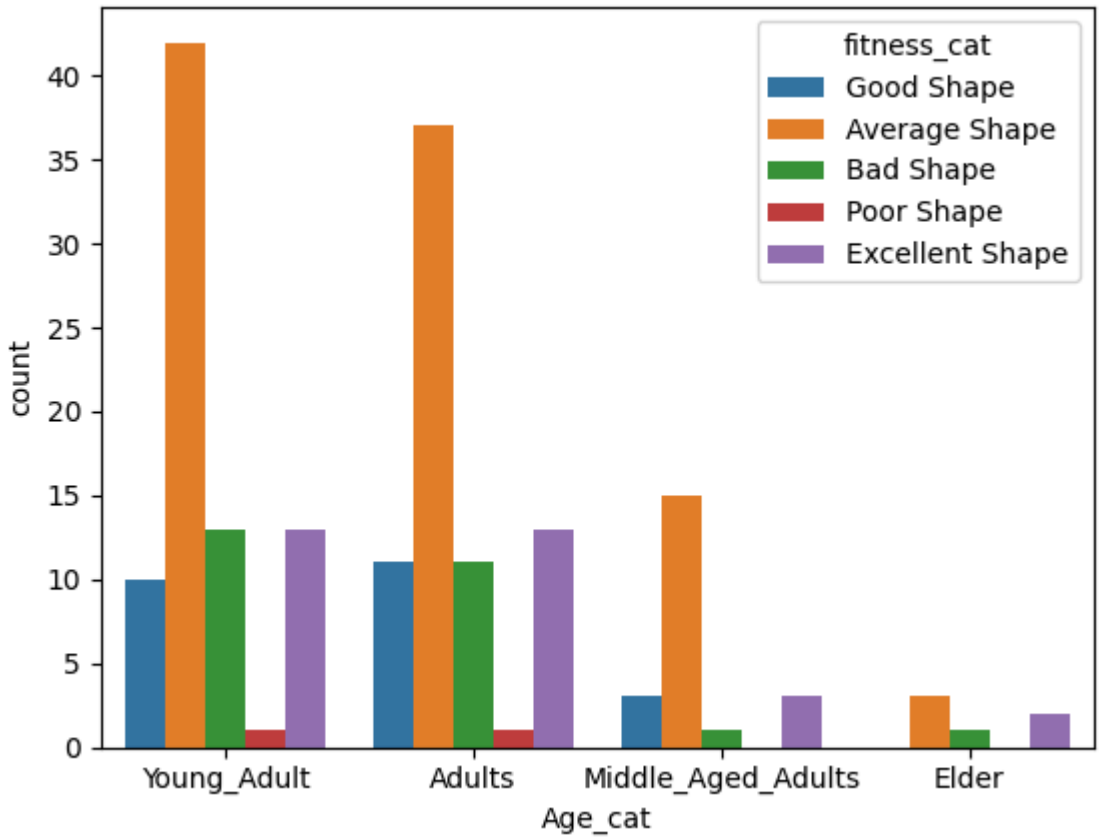
In [ ]:
```
sns.violinplot(data=data, y=data.Product, x=data.Age_cat, hue=data.Gender)
```

Out[ ]:
```
<Axes: xlabel='Age_cat', ylabel='Product'>
```



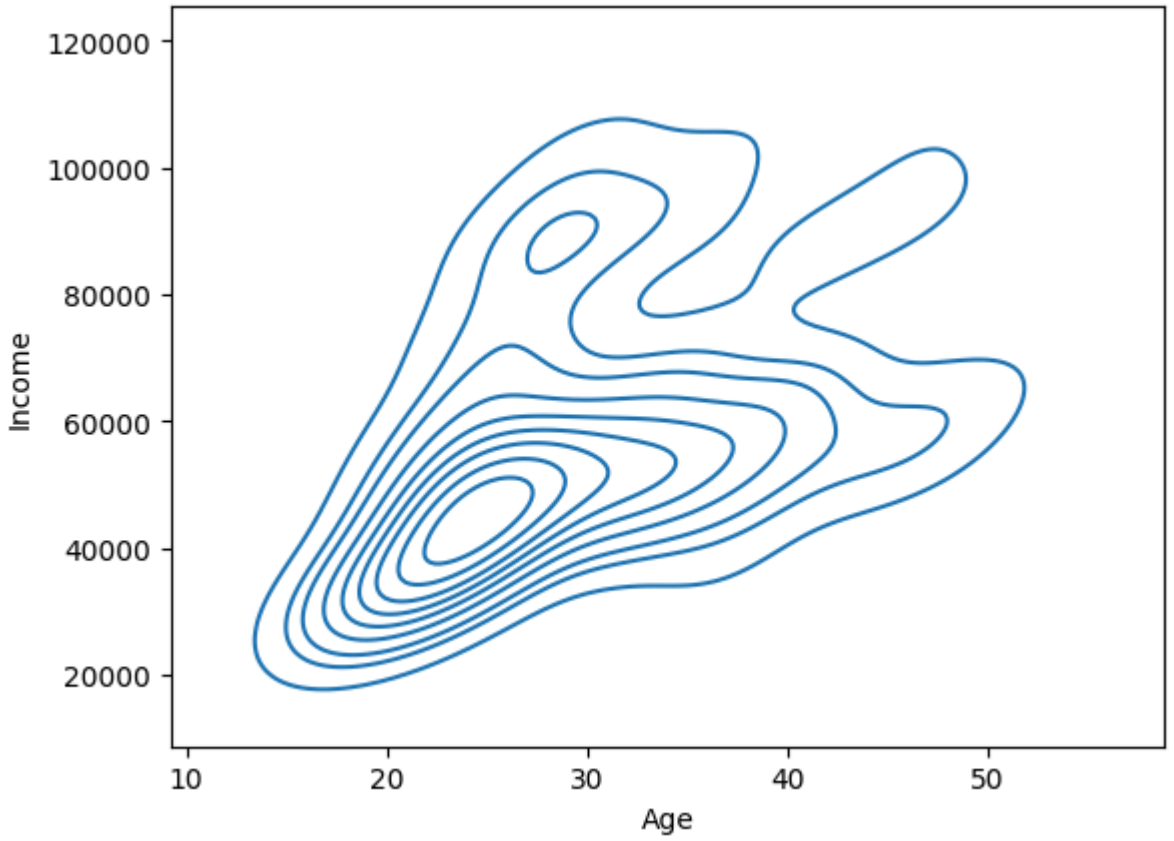
In [ ]:
```
sns.countplot(data=data, x=data.Age_cat, hue=data.fitness_cat)
```

Out[ ]:
```
<Axes: xlabel='Age_cat', ylabel='count'>
```



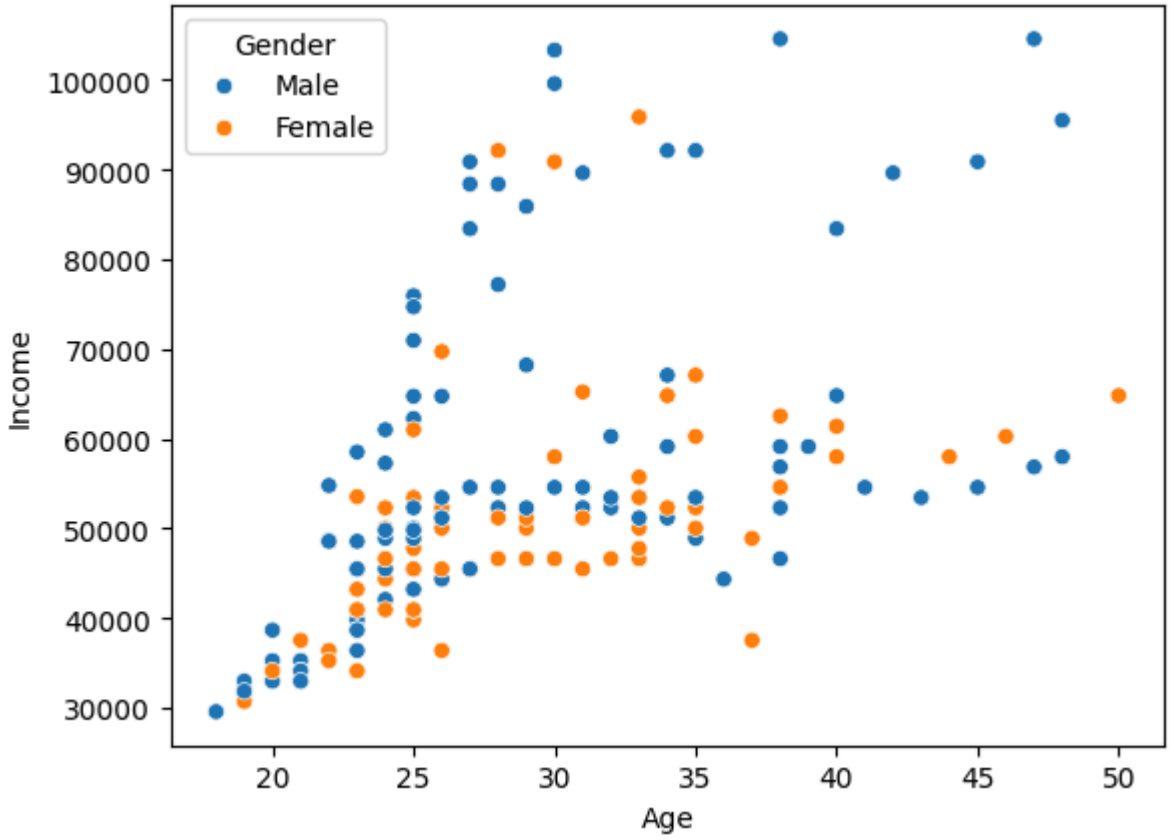
In [ ]:
```
sns.kdeplot(data=data, x=data.Age, y=data.Income)
```

Out[ ]:
```
<Axes: xlabel='Age', ylabel='Income'>
```



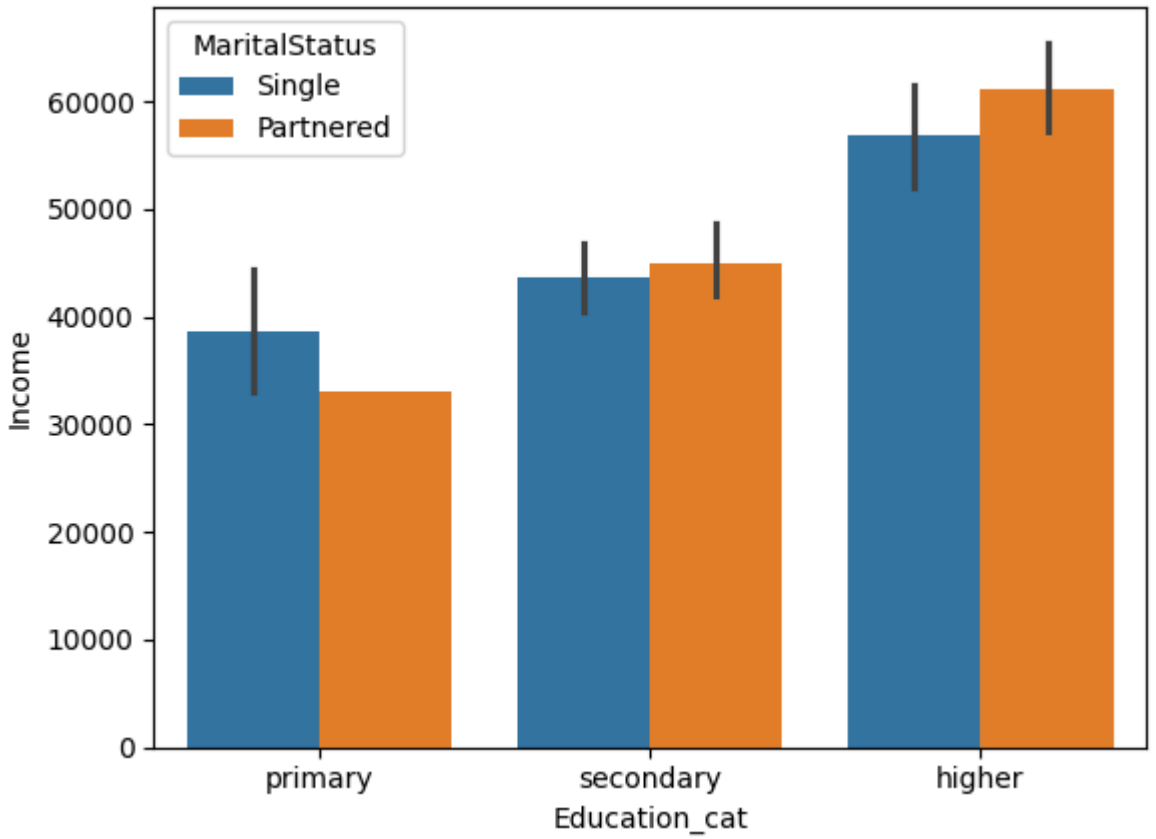
In [ ]:
```
sns.scatterplot(data=data, x=data.Age, y=data.Income, hue=data.Gender)
```

Out[ ]: <Axes: xlabel='Age', ylabel='Income'>



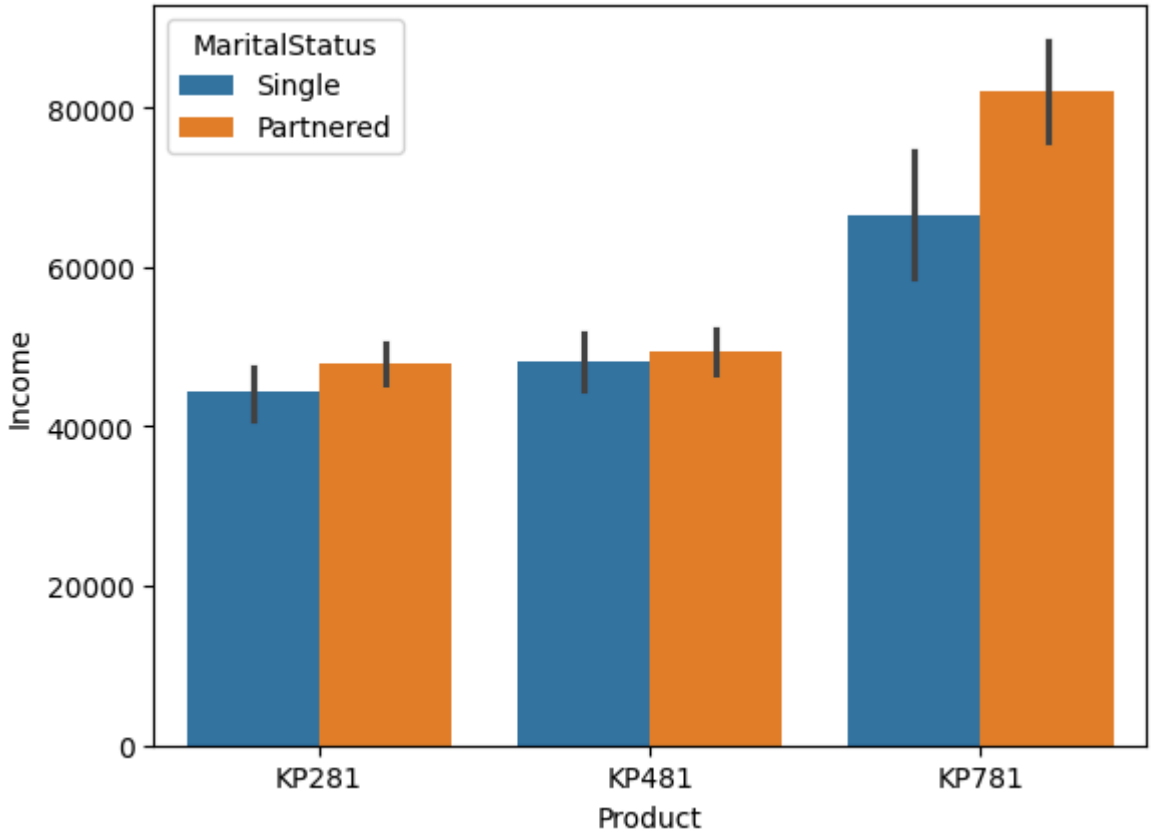
In [ ]:
```
sns.barplot(data=data, x= data.Education_cat, y=data.Income, hue=data.MaritalStatus)
```

Out[ ]:
```
<Axes: xlabel='Education_cat', ylabel='Income'>
```



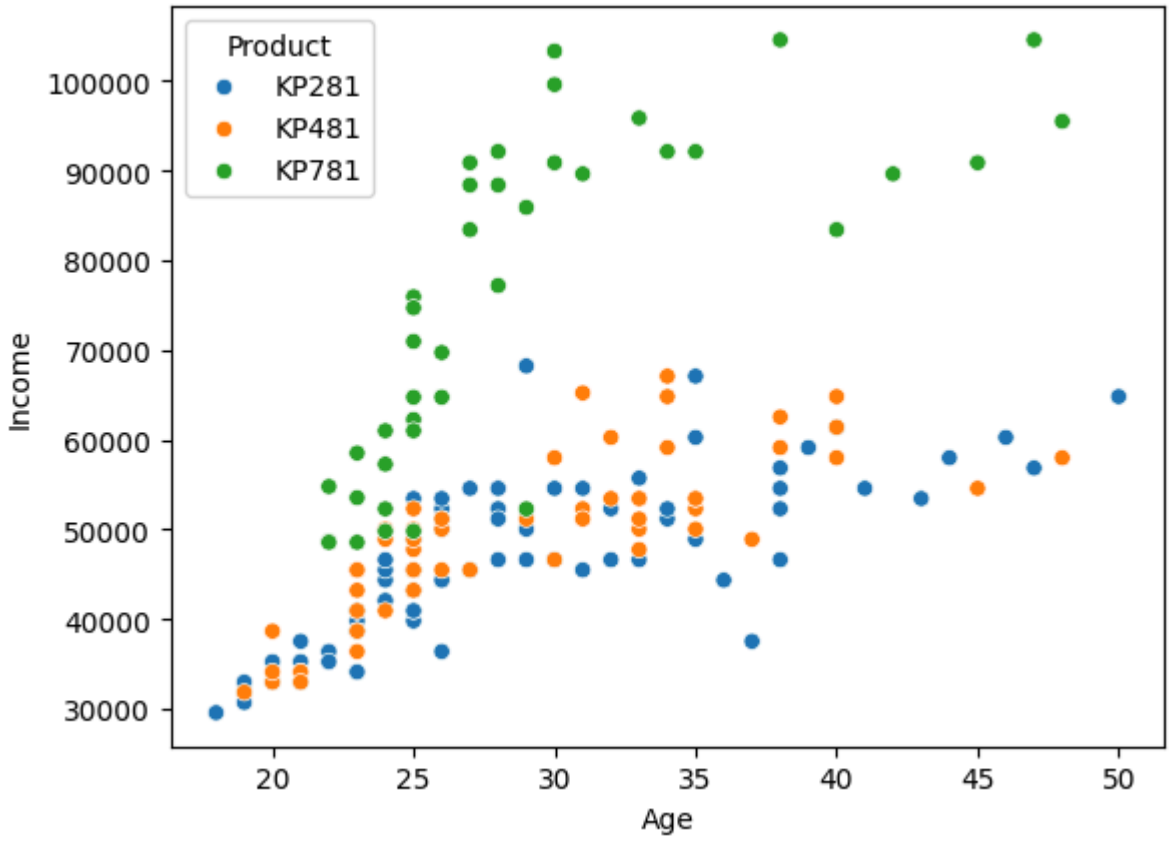
In [ ]:
```
sns.barplot(data=data, x= data.Product, y=data.Income, hue=data.MaritalStatus)
```

Out[ ]:
```
<Axes: xlabel='Product', ylabel='Income'>
```



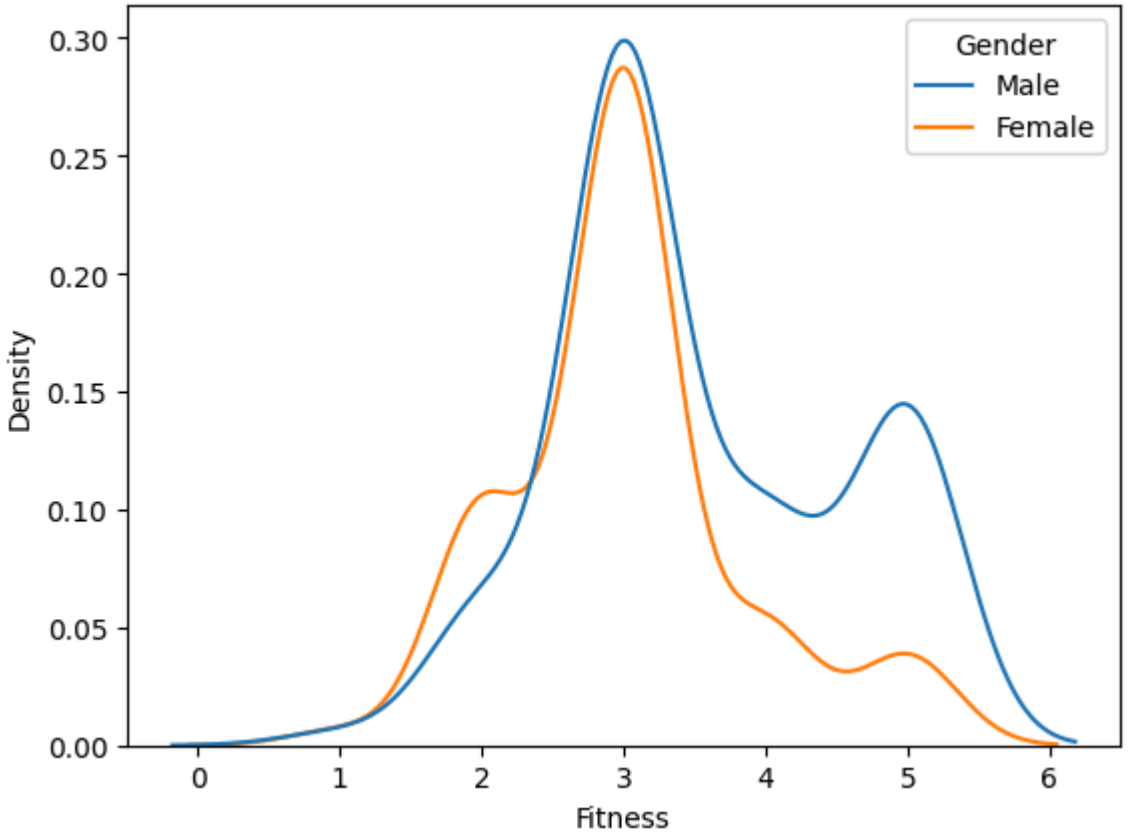
In [ ]:
```
sns.scatterplot(data=data, x= data.Age, y=data.Income, hue=data.Product)
```

Out[ ]:
```
<Axes: xlabel='Age', ylabel='Income'>
```



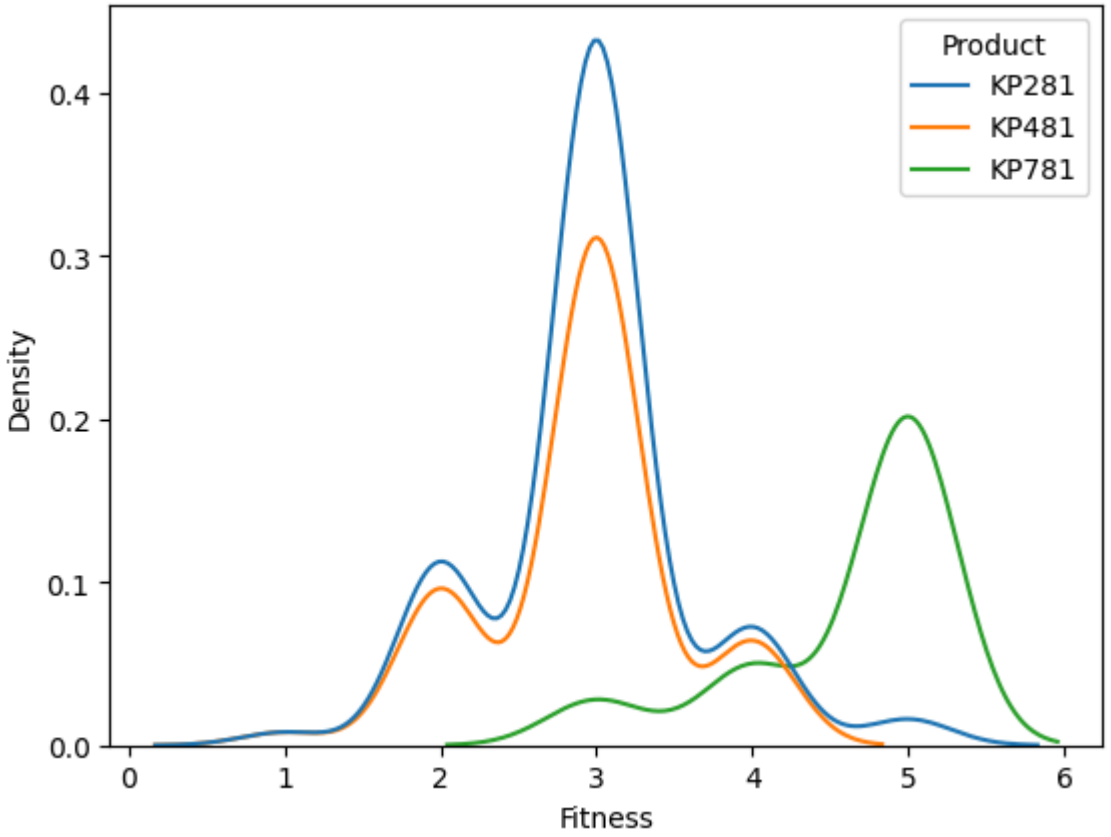
In [ ]:
```
sns.kdeplot(data=data, x=data.Fitness, hue=data.Gender)
```

Out[ ]:
```
<Axes: xlabel='Fitness', ylabel='Density'>
```



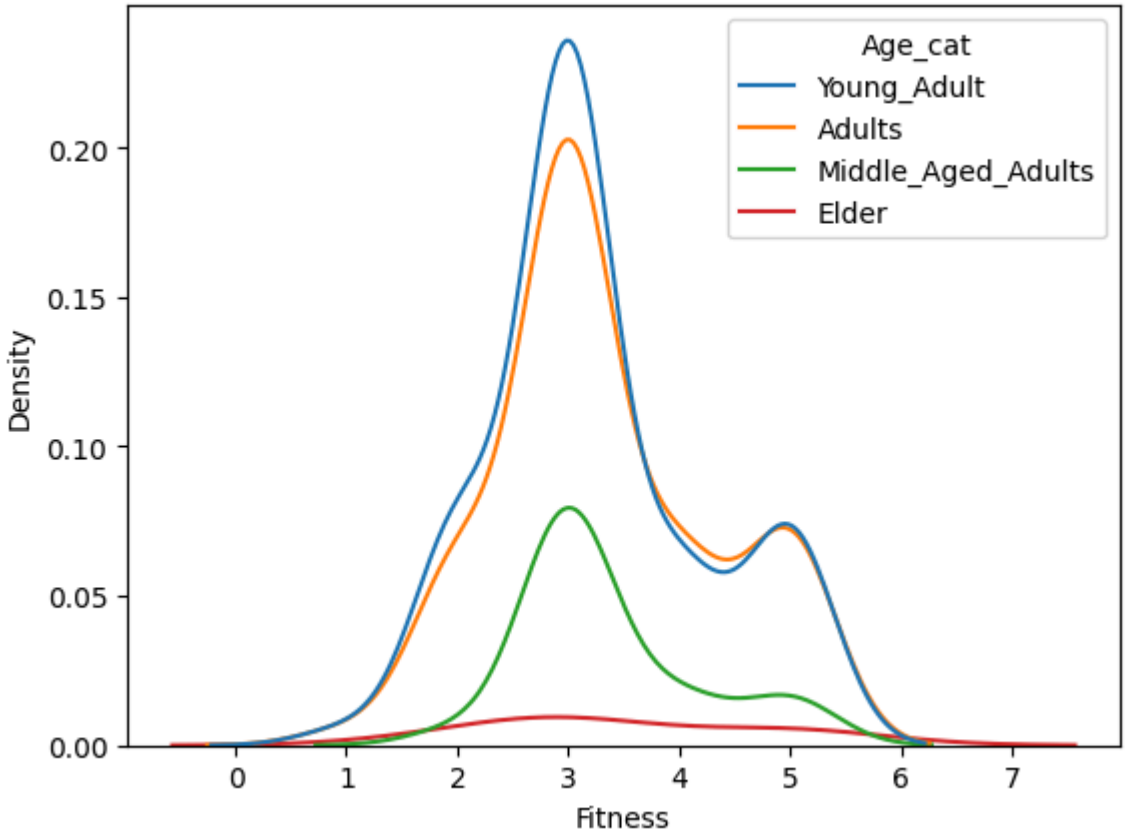
In [ ]:
```
sns.kdeplot(data=data, x=data.Fitness, hue=data.Product)
```

Out[ ]:
```
<Axes: xlabel='Fitness', ylabel='Density'>
```



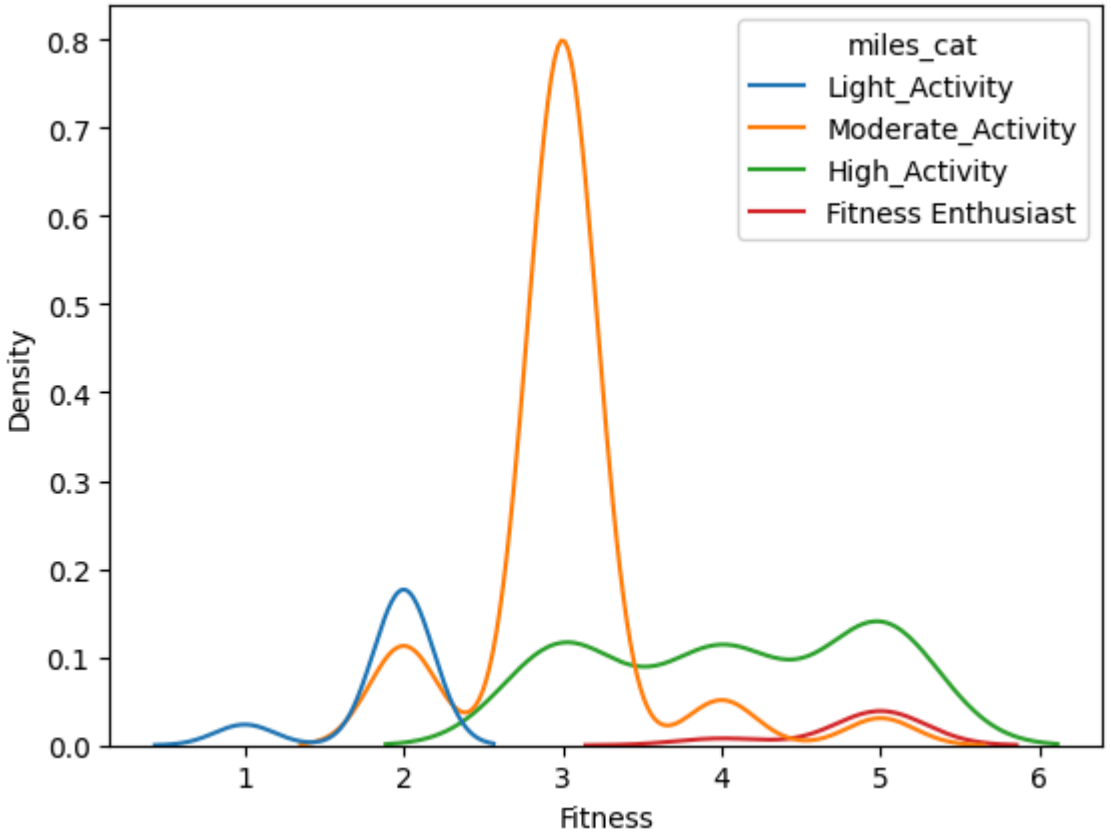
In [ ]:
```
sns.kdeplot(data=data, x=data.Fitness, hue=data.Age_cat)
```

Out[ ]:
```
<Axes: xlabel='Fitness', ylabel='Density'>
```



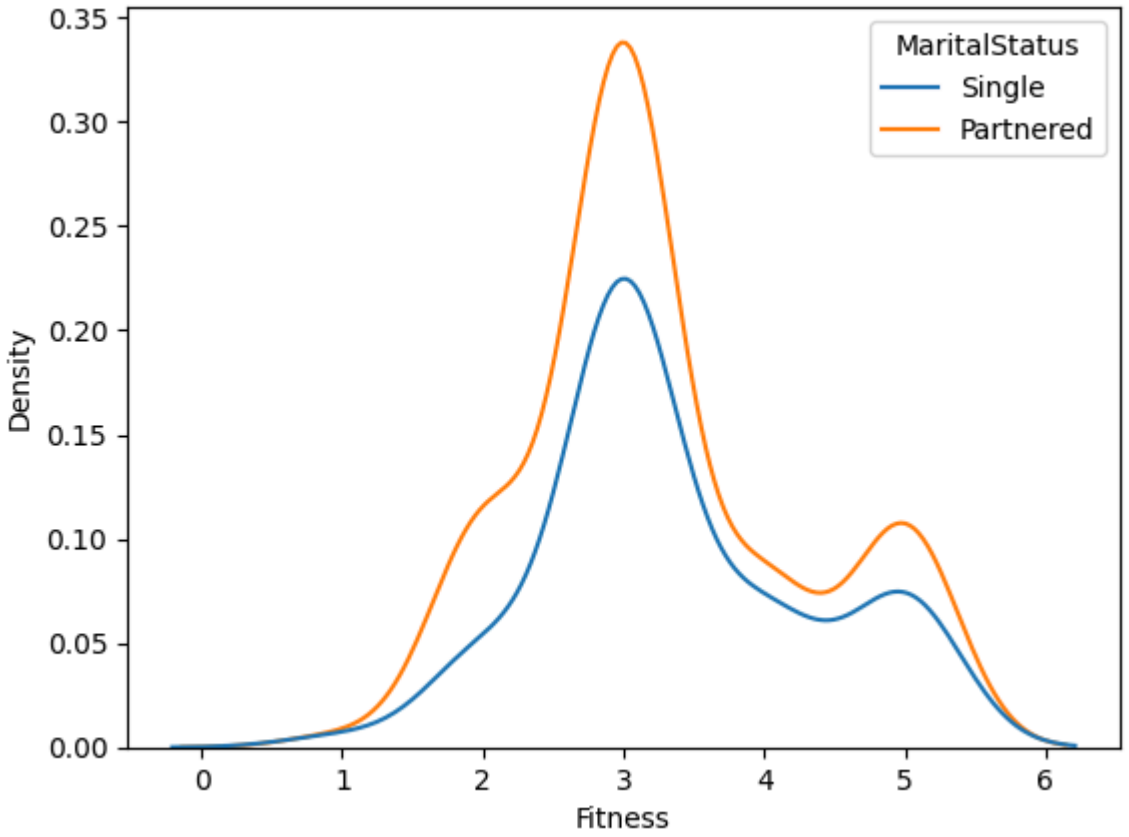
In [ ]:
```
sns.kdeplot(data=data, x=data.Fitness, hue=data.miles_cat)
```

Out[ ]:
```
<Axes: xlabel='Fitness', ylabel='Density'>
```



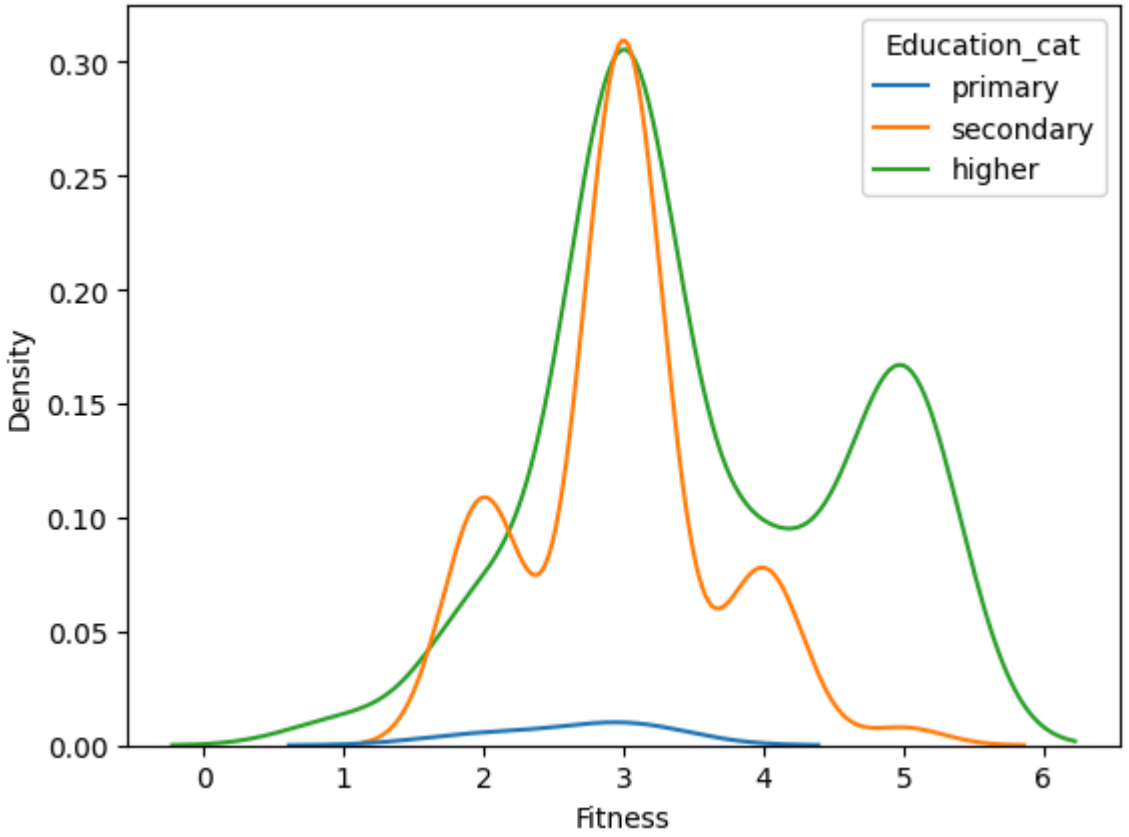
In [ ]:
```
sns.kdeplot(data=data, x=data.Fitness, hue=data.MaritalStatus)
```

Out[ ]:
```
<Axes: xlabel='Fitness', ylabel='Density'>
```



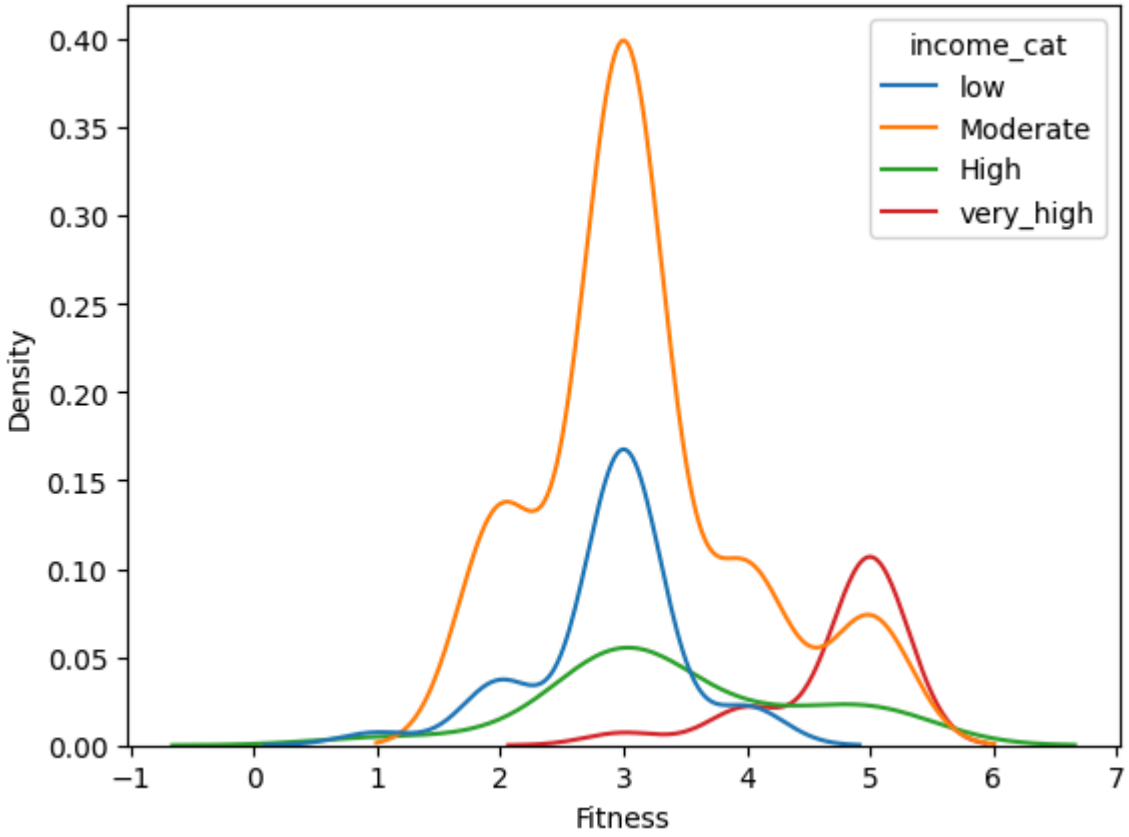
In [ ]:
```
sns.kdeplot(data=data, x=data.Fitness, hue=data.Education_cat)
```

Out[ ]:
```
<Axes: xlabel='Fitness', ylabel='Density'>
```

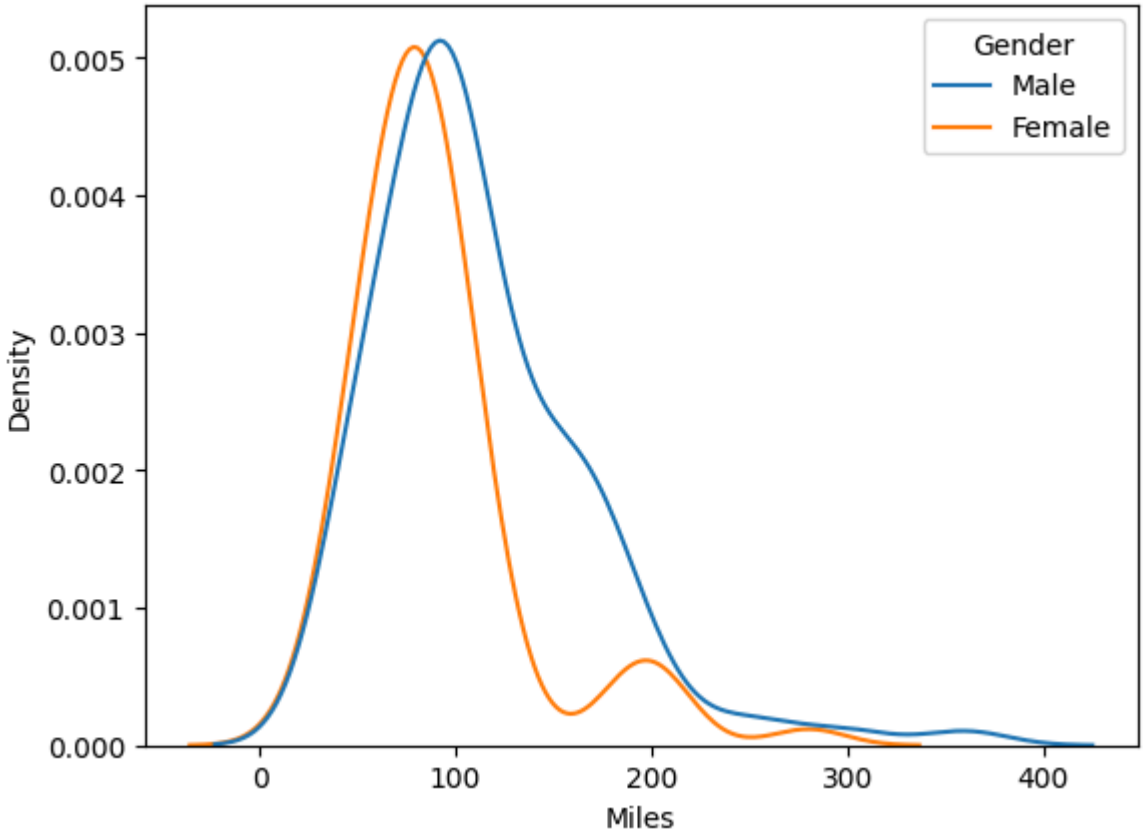


In [ ]:
```
sns.kdeplot(data=data, x=data.Fitness, hue=data.income_cat)
```

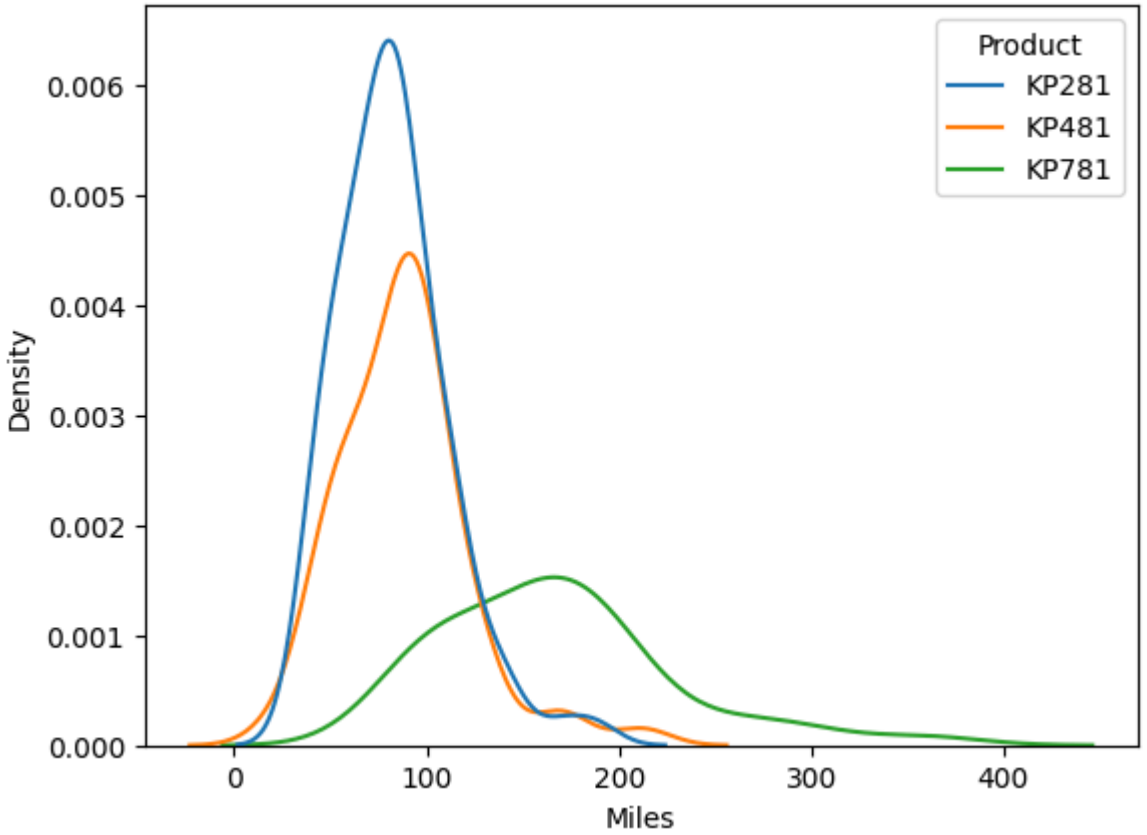
Out[ ]:
```
<Axes: xlabel='Fitness', ylabel='Density'>
```



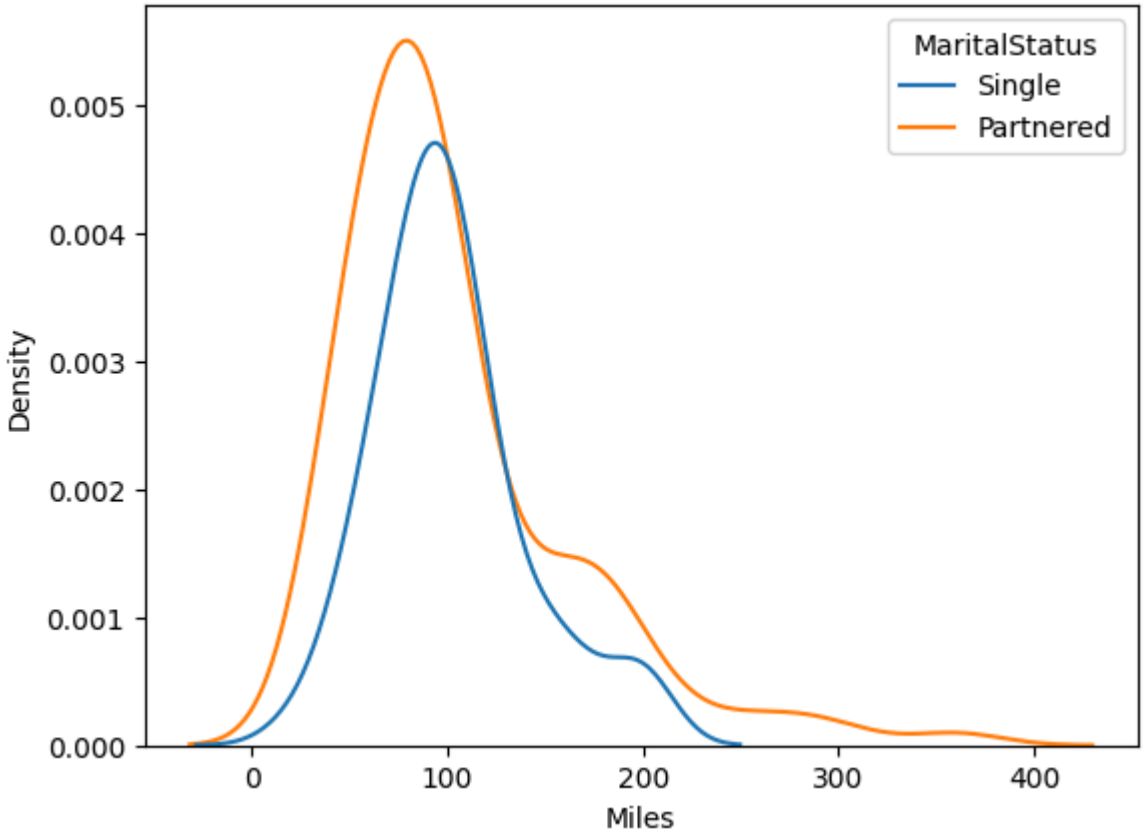
In [ ]:
```
sns.kdeplot(data=data, x=data.Miles, hue=data.Gender)
plt.show()
```



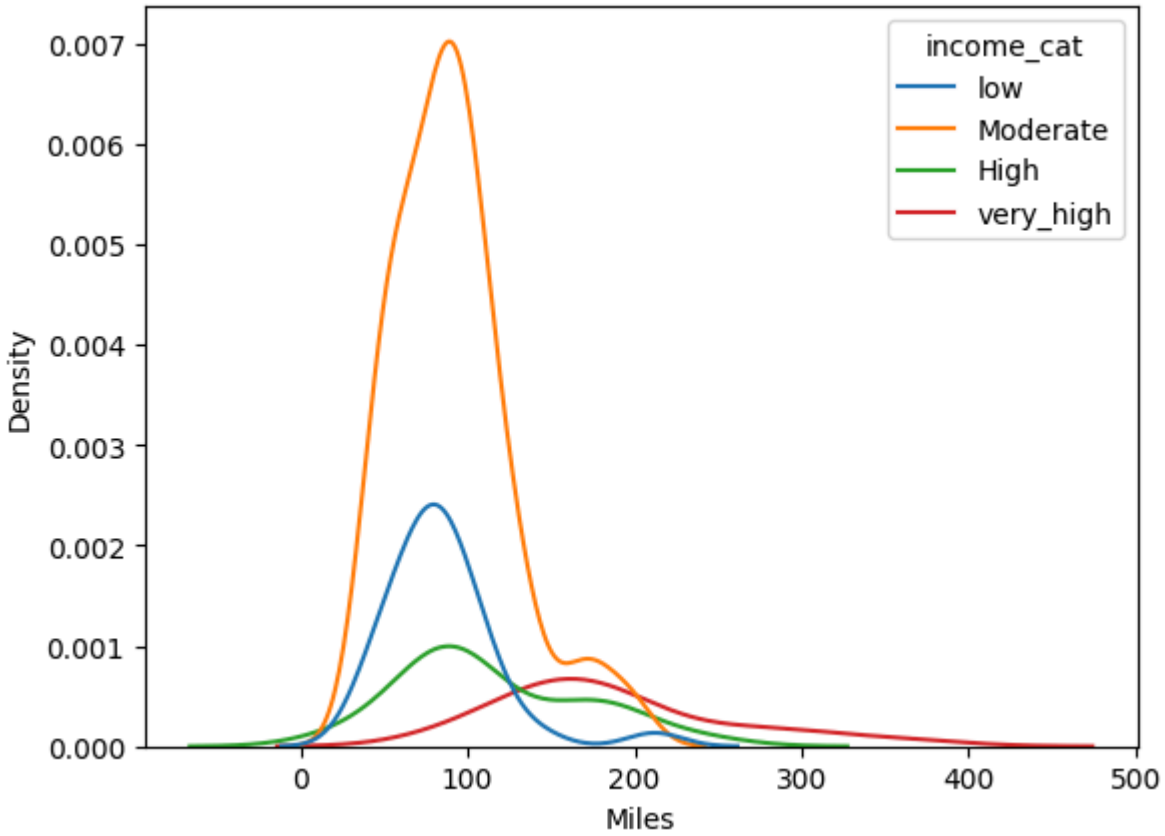
In [ ]:
```
sns.kdeplot(data=data, x=data.Miles, hue=data.Product)
plt.show()
```



In [ ]:
```
sns.kdeplot(data=data, x=data.Miles, hue=data.MaritalStatus)
plt.show()
```

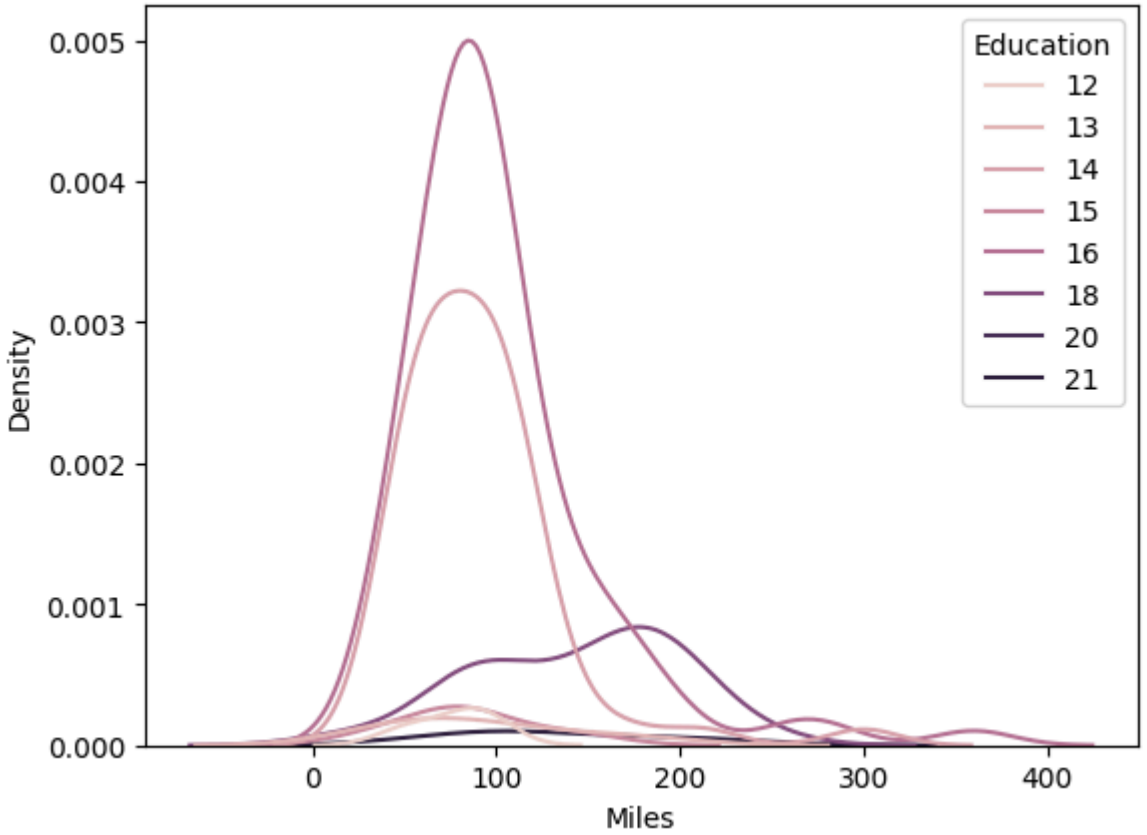


In [ ]:
```
sns.kdeplot(data=data, x=data.Miles, hue=data.income_cat)
plt.show()
```

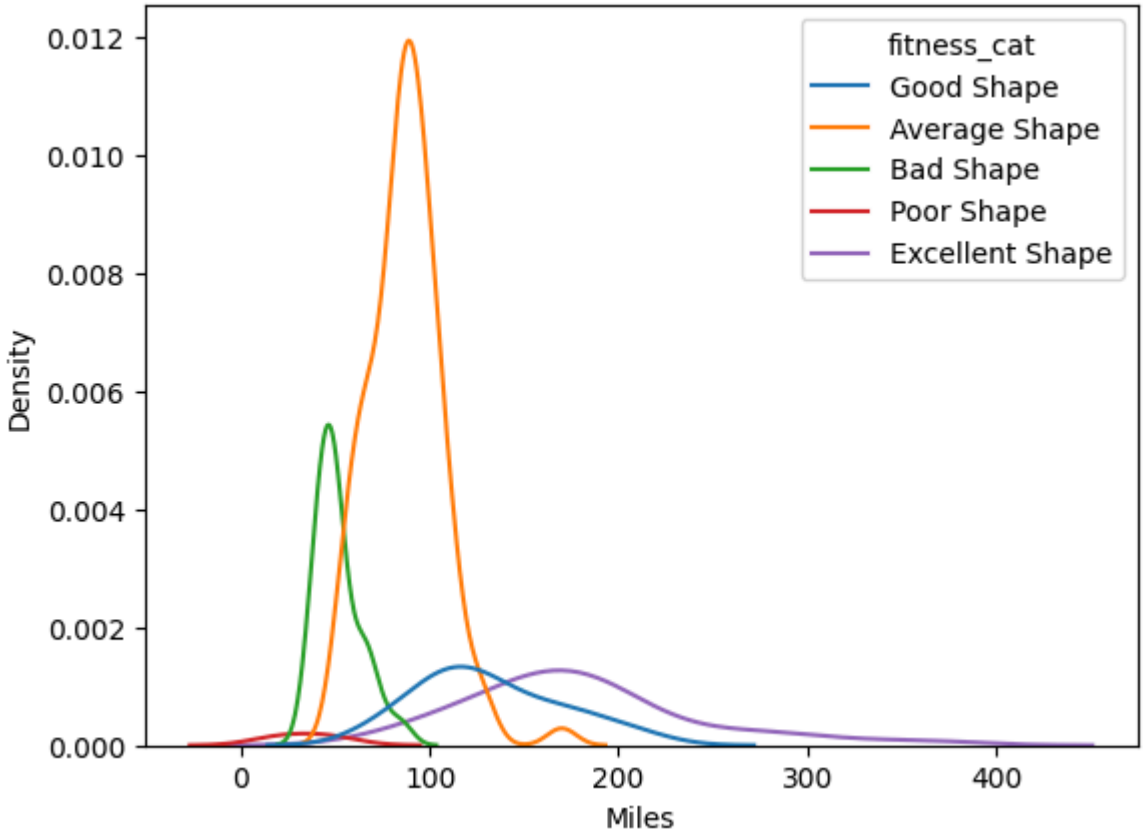


In [ ]:
```
sns.kdeplot(data=data, x=data.Miles, hue=data.Education)
plt.show()
```

```
<ipython-input-66-7112b4683212>:1: UserWarning: Dataset has 0 variance; skipping density estimate. Pass `warn_singul
ar=False` to disable this warning.
  sns.kdeplot(data=data, x=data.Miles, hue=data.Education)
```

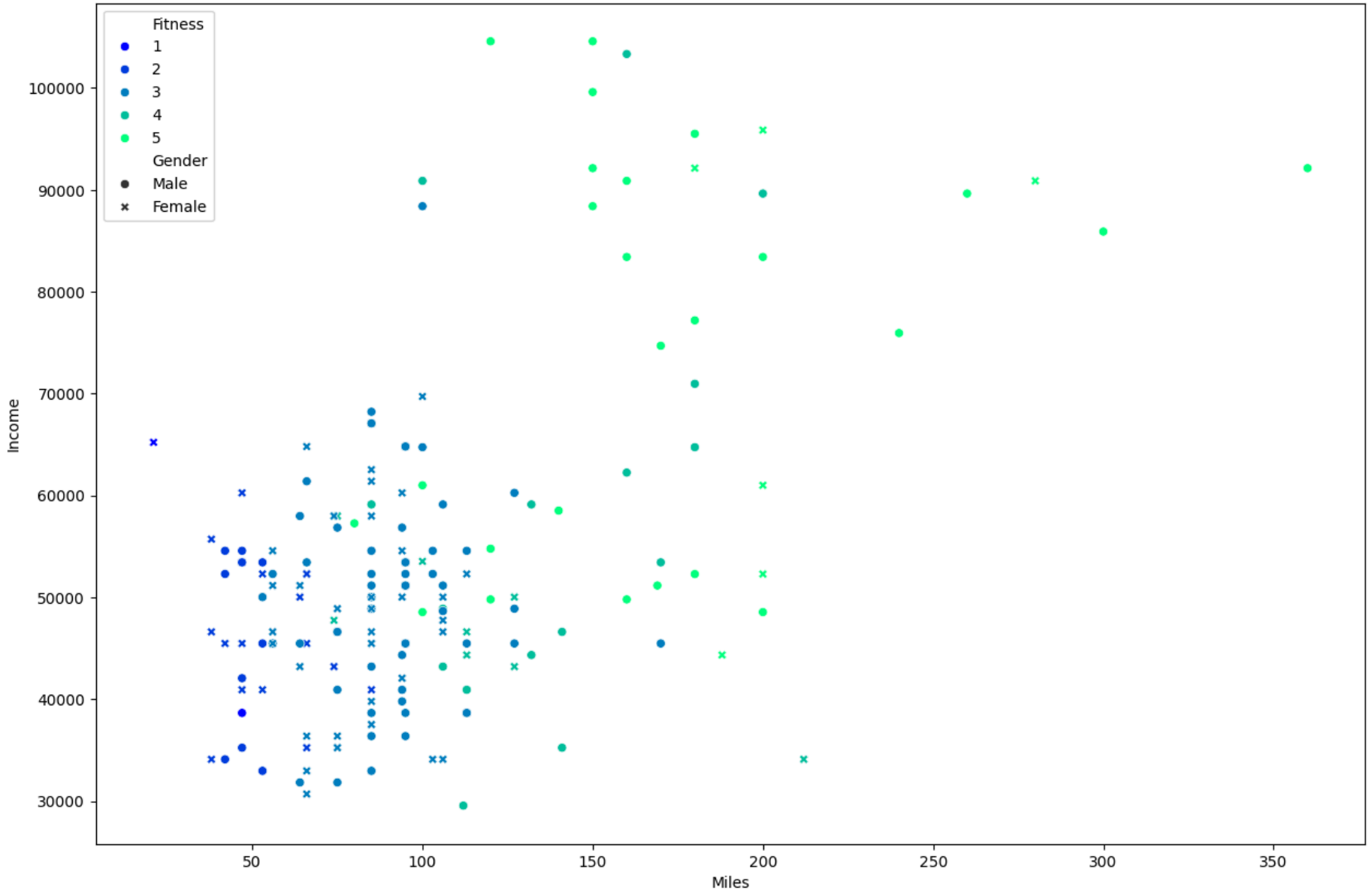


In [ ]:
```
sns.kdeplot(data=data, x=data.Miles, hue=data.fitness_cat)
plt.show()
```

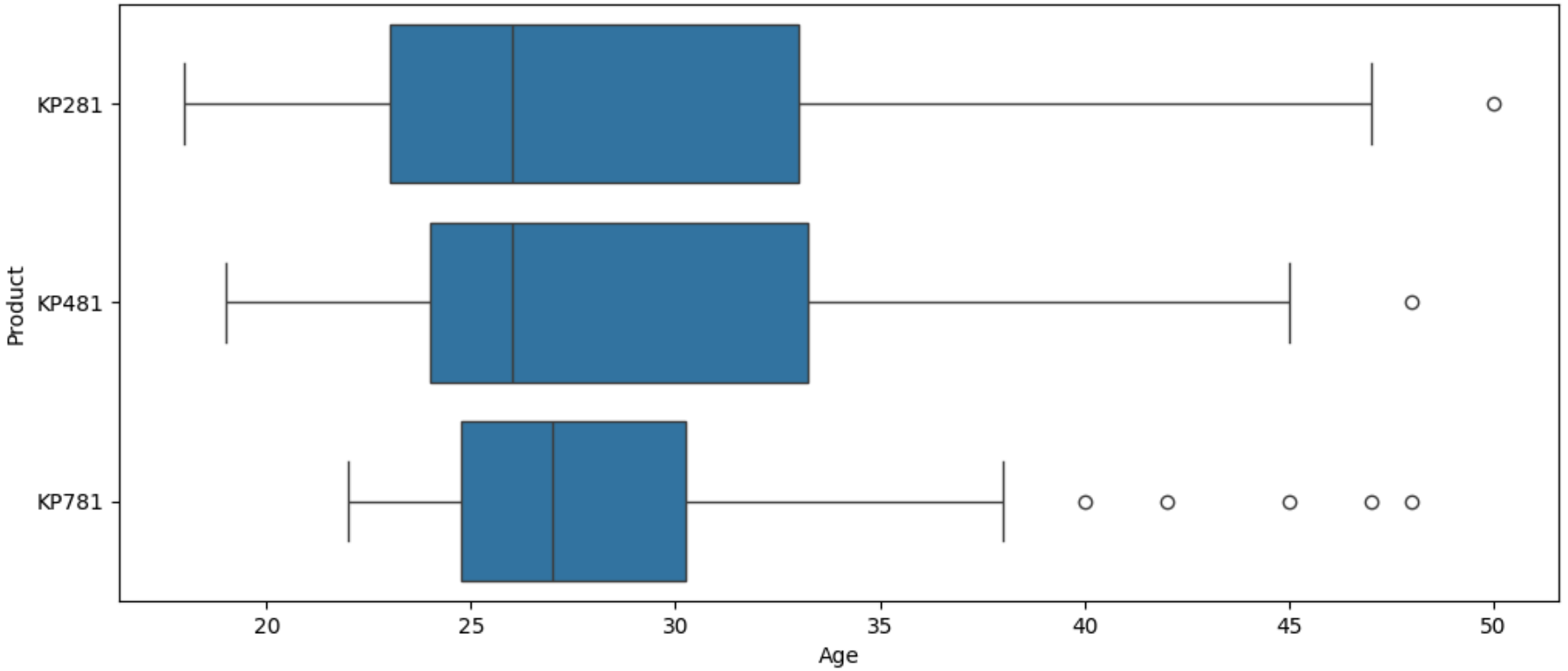


In [ ]:
```
# Scatter Plot
plt.figure(figsize=(15,10))
sns.scatterplot(x='Miles',y='Income',data=data,hue='Fitness',style='Gender',palette='winter')
```

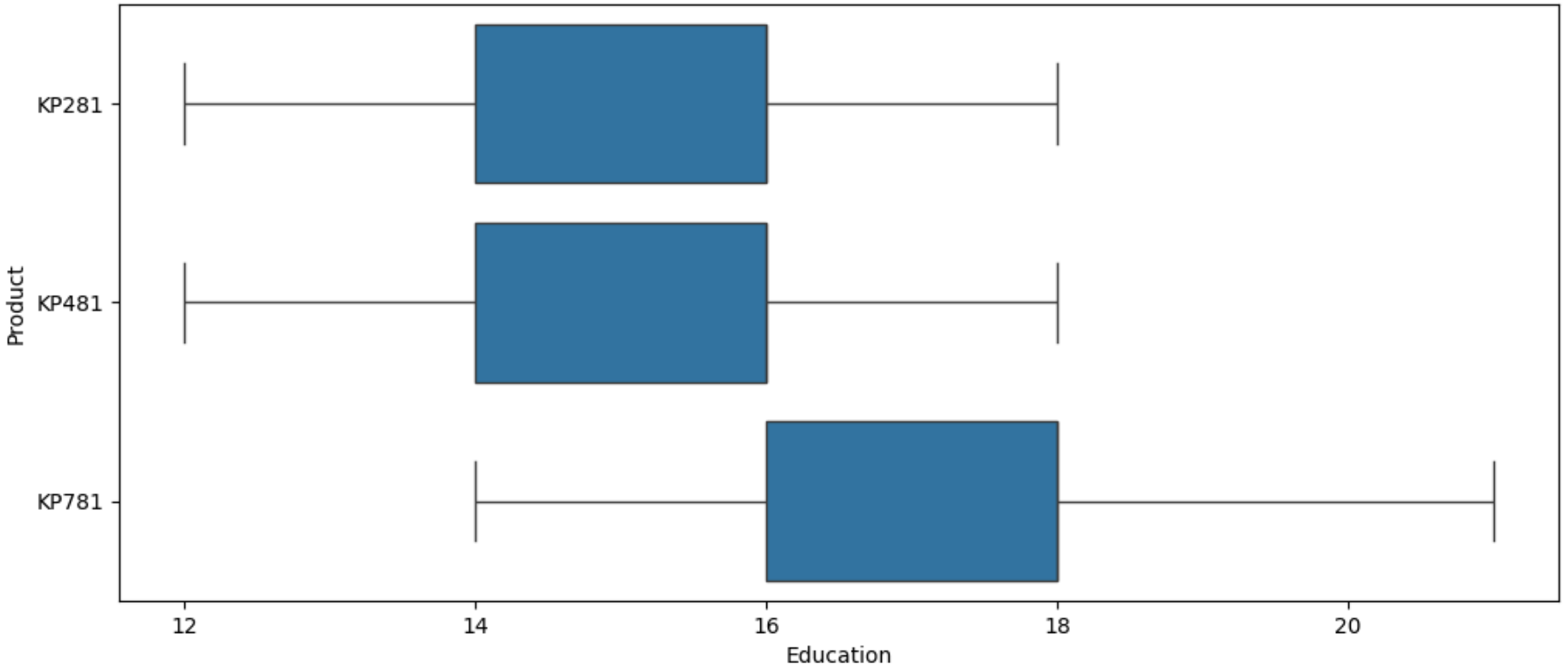
Out[ ]: <Axes: xlabel='Miles', ylabel='Income'>



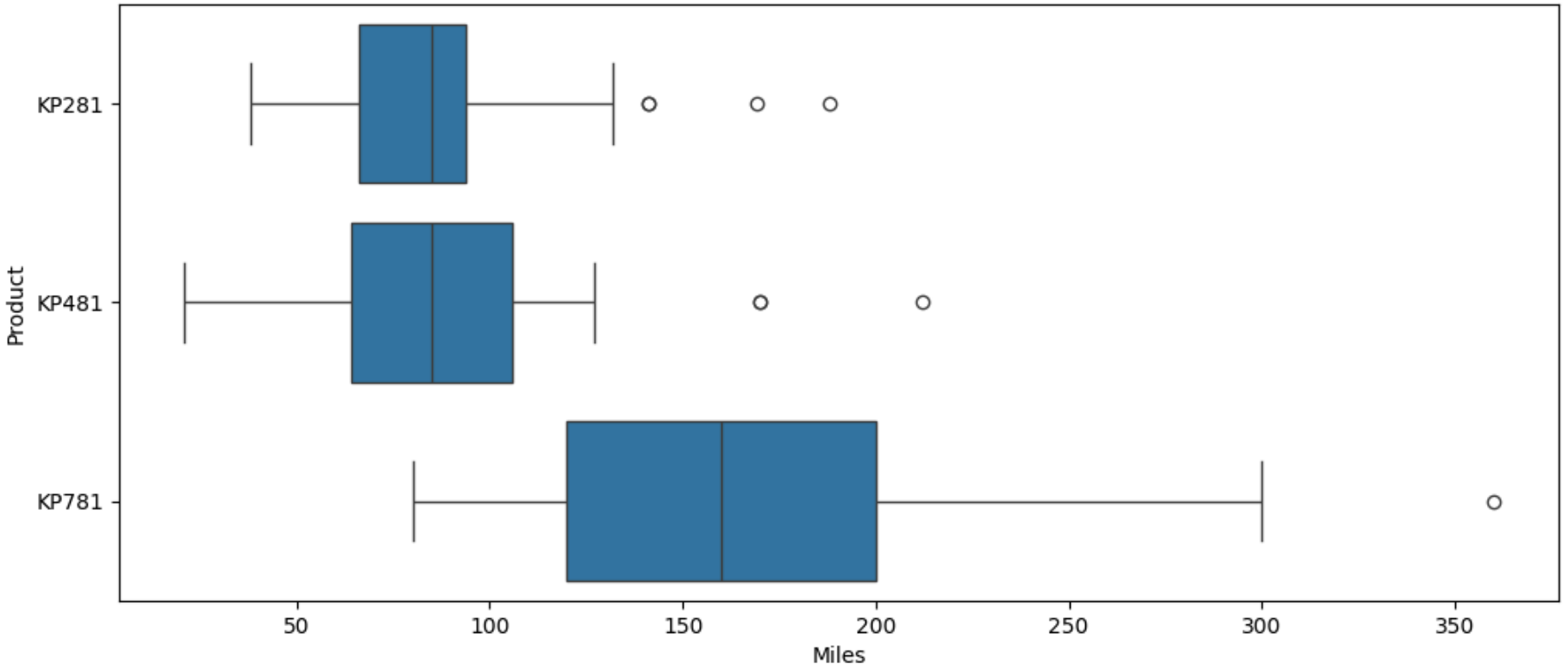
In [ ]:
```
plt.figure(figsize=(12,5))
sns.boxplot(data= data, y=data.Product, x=data.Age)
plt.show()
```



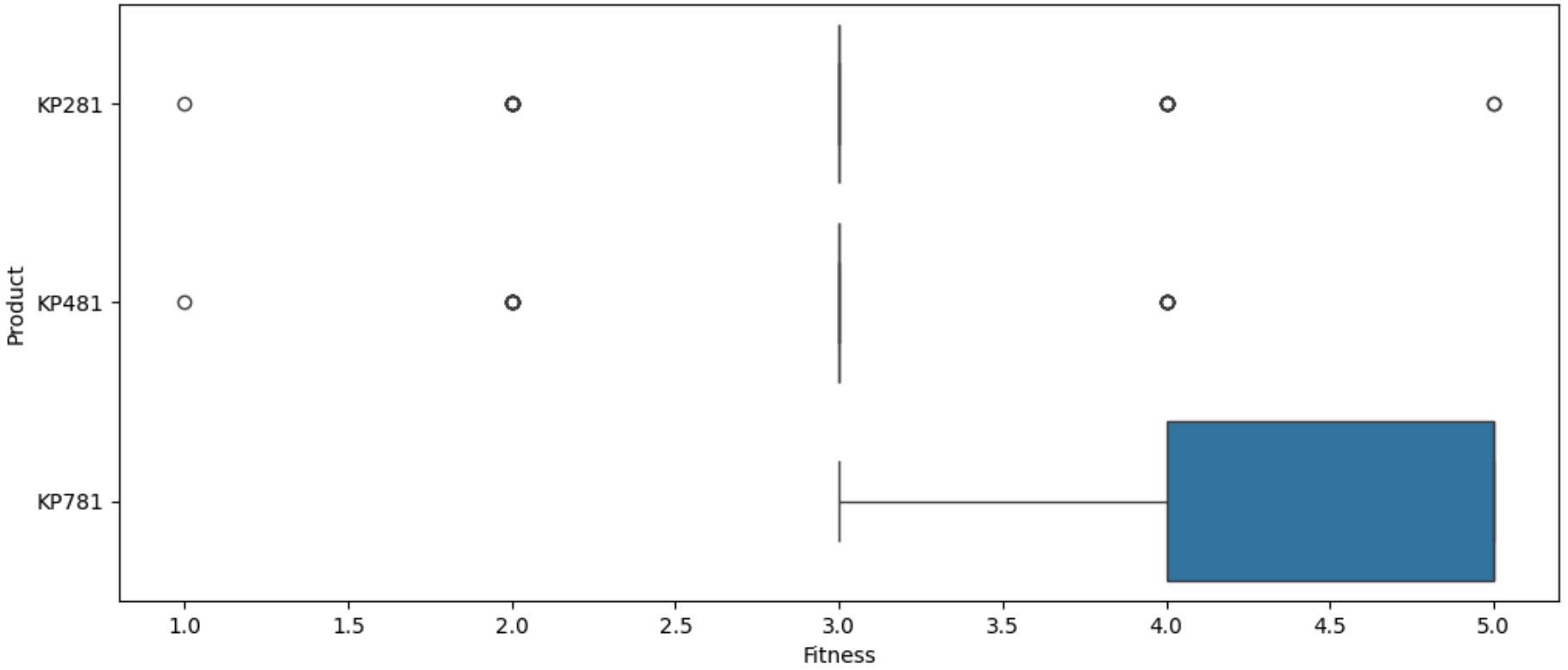
In [ ]:
```
plt.figure(figsize=(12,5))
sns.boxplot(data= data, y=data.Product, x=data.Education)
plt.show()
```



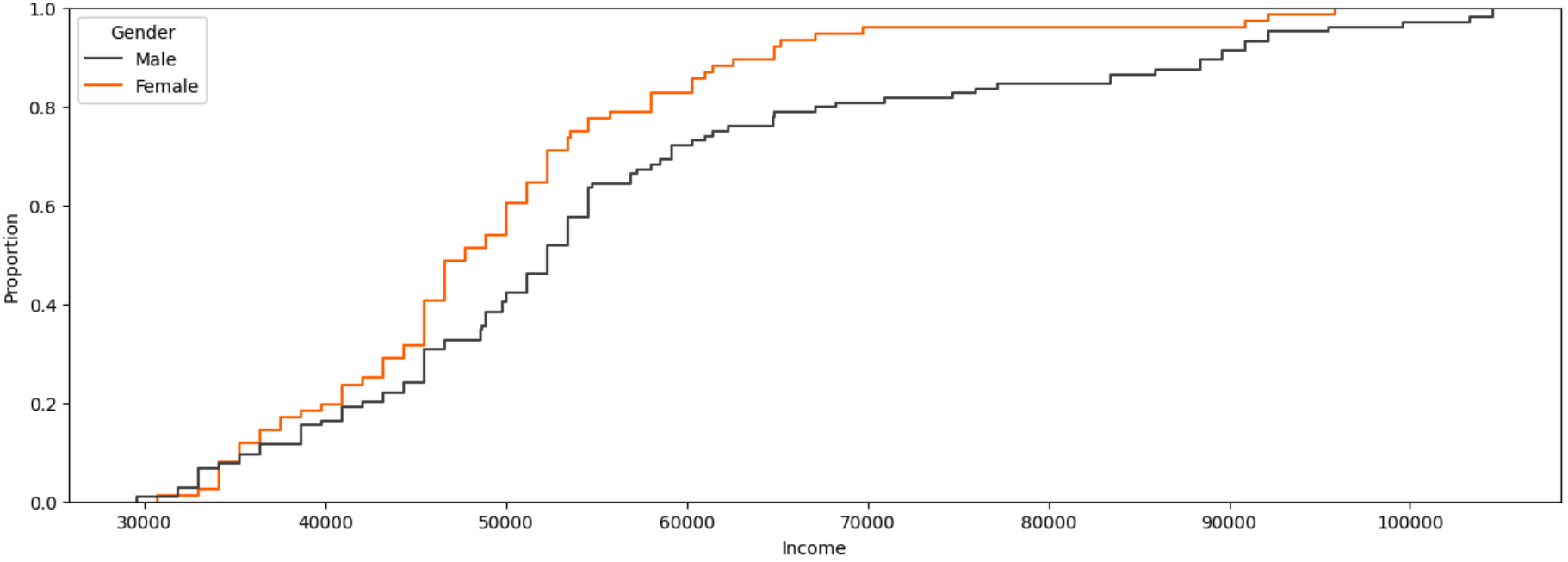
In [ ]:
```
plt.figure(figsize=(12,5))
sns.boxplot(data= data, y=data.Product, x=data.Miles)
plt.show()
```



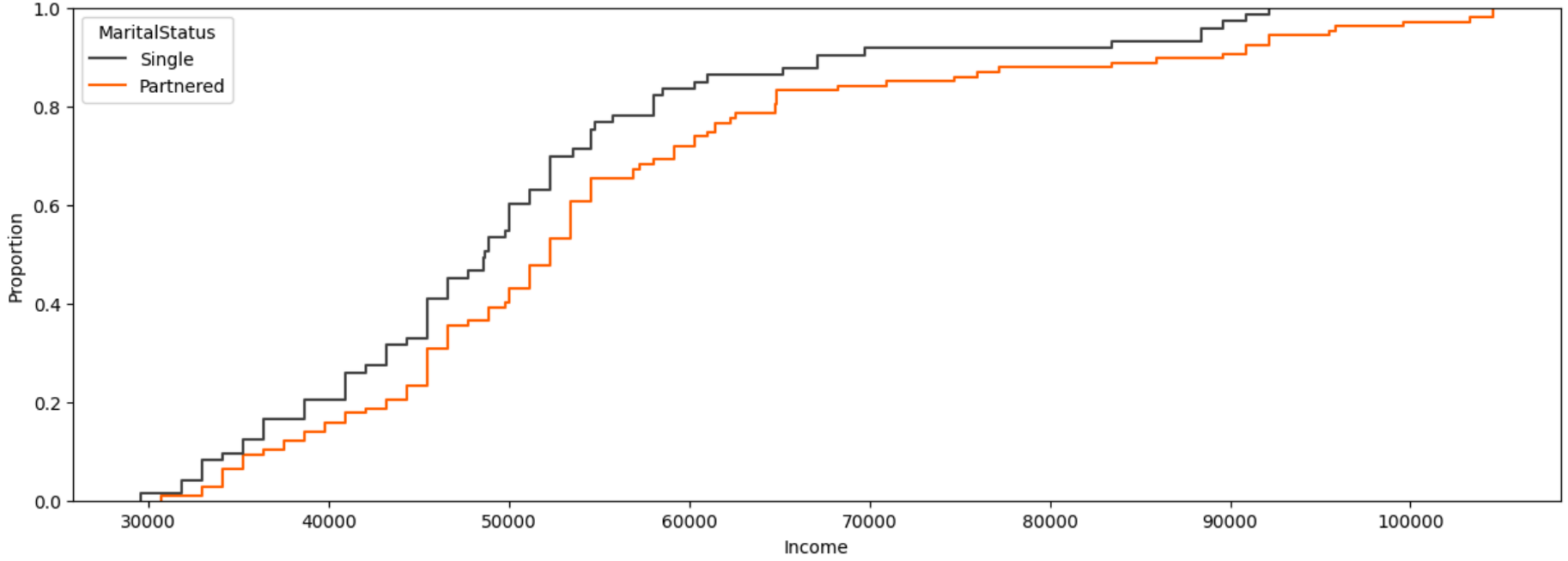
In [ ]:
```
plt.figure(figsize=(12,5))
sns.boxplot(data= data, y=data.Product, x=data.Fitness)
plt.show()
```



In [ ]:
```
# Empirical Cumulative Distribution Function - proportional distribution for Income of customers against their Gende
plt.figure(figsize=(15,5))
sns.ecdfplot(data=data,x='Income',hue='Gender',complementary=False,palette=['#454545','#FF6000'])
plt.show()
```

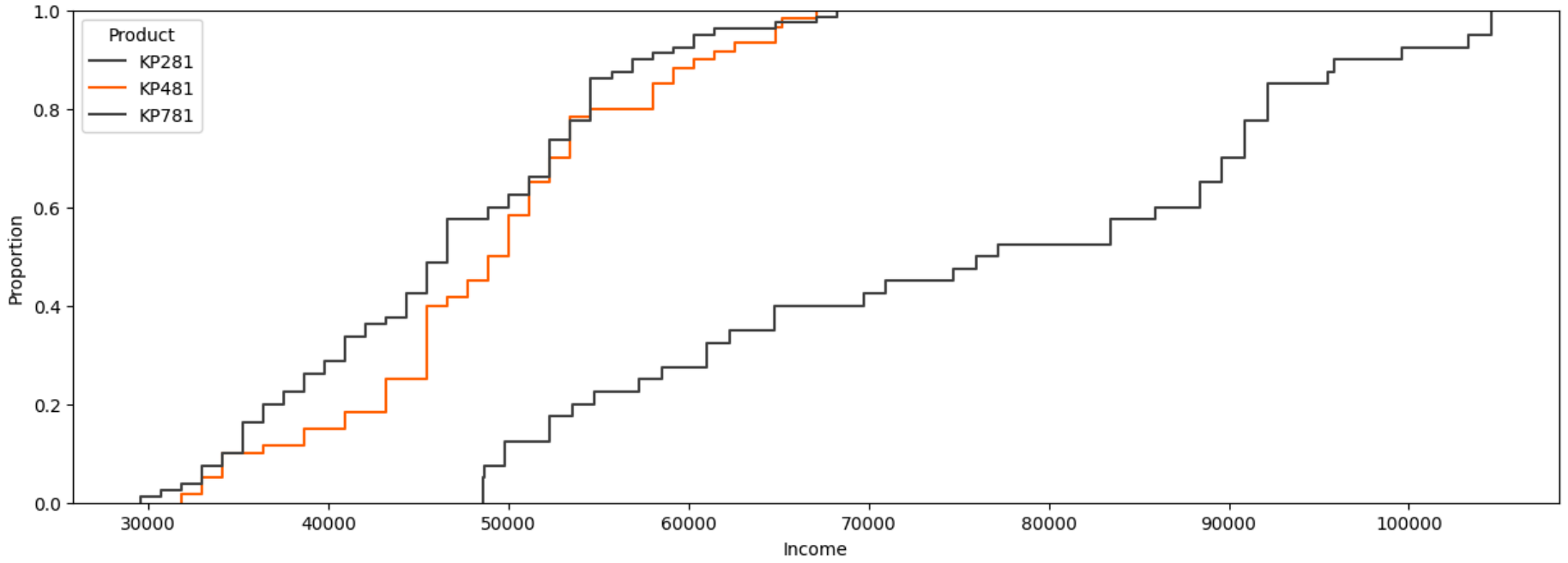


In [ ]:
```
# Empirical Cumulative Distribution Function - proportional distribution for Income of customers against their Gende
plt.figure(figsize=(15,5))
sns.ecdfplot(data=data, x='Income',hue='MaritalStatus',complementary=False,palette=['#454545','#FF6000'])
plt.show()
```



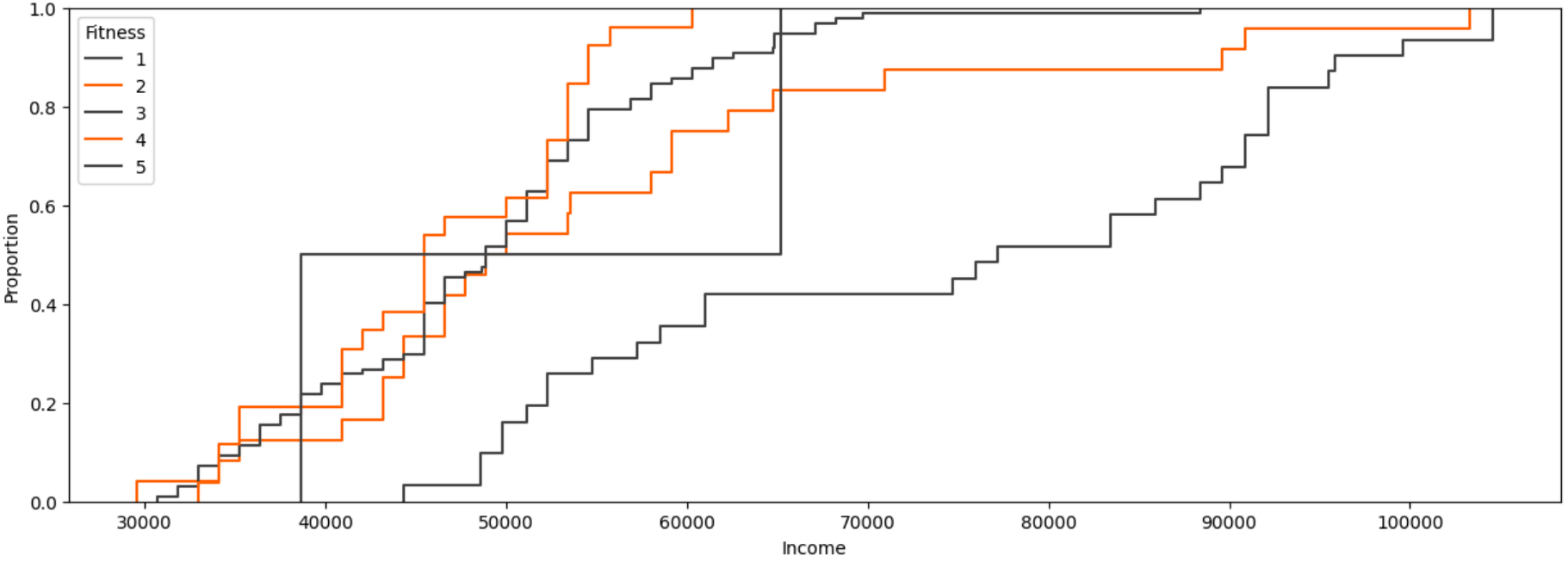
In [ ]:
```
plt.figure(figsize=(15,5))
sns.ecdfplot(data=data, x='Income',hue='Product',complementary=False,palette=['#454545','#FF6000'])
plt.show()
```

```
<ipython-input-75-b1dda023dec8>:2: UserWarning:
The palette list has fewer values (2) than needed (3) and will cycle, which may produce an uninterpretable plot.
  sns.ecdfplot(data=data, x='Income',hue='Product',complementary=False,palette=['#454545','#FF6000'])
```



In [ ]:
```
plt.figure(figsize=(15,5))
sns.ecdfplot(data=data, x='Income',hue='Fitness',complementary=False,palette=['#454545','#FF6000'])
plt.show()
```

```
<ipython-input-76-822a178138a0>:2: UserWarning:
The palette list has fewer values (2) than needed (5) and will cycle, which may produce an uninterpretable plot.
  sns.ecdfplot(data=data, x='Income',hue='Fitness',complementary=False,palette=['#454545','#FF6000'])
```

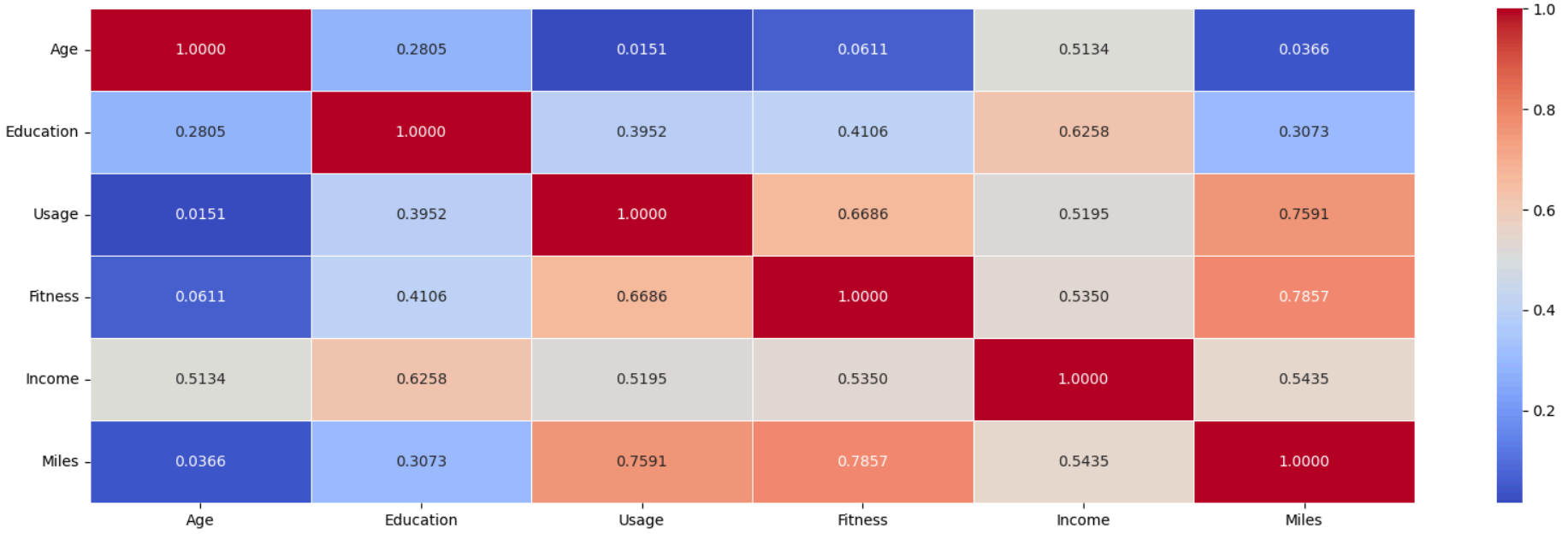


In [ ]:
```
data_corr = data.select_dtypes([float,int])
data_corr.corr()
```

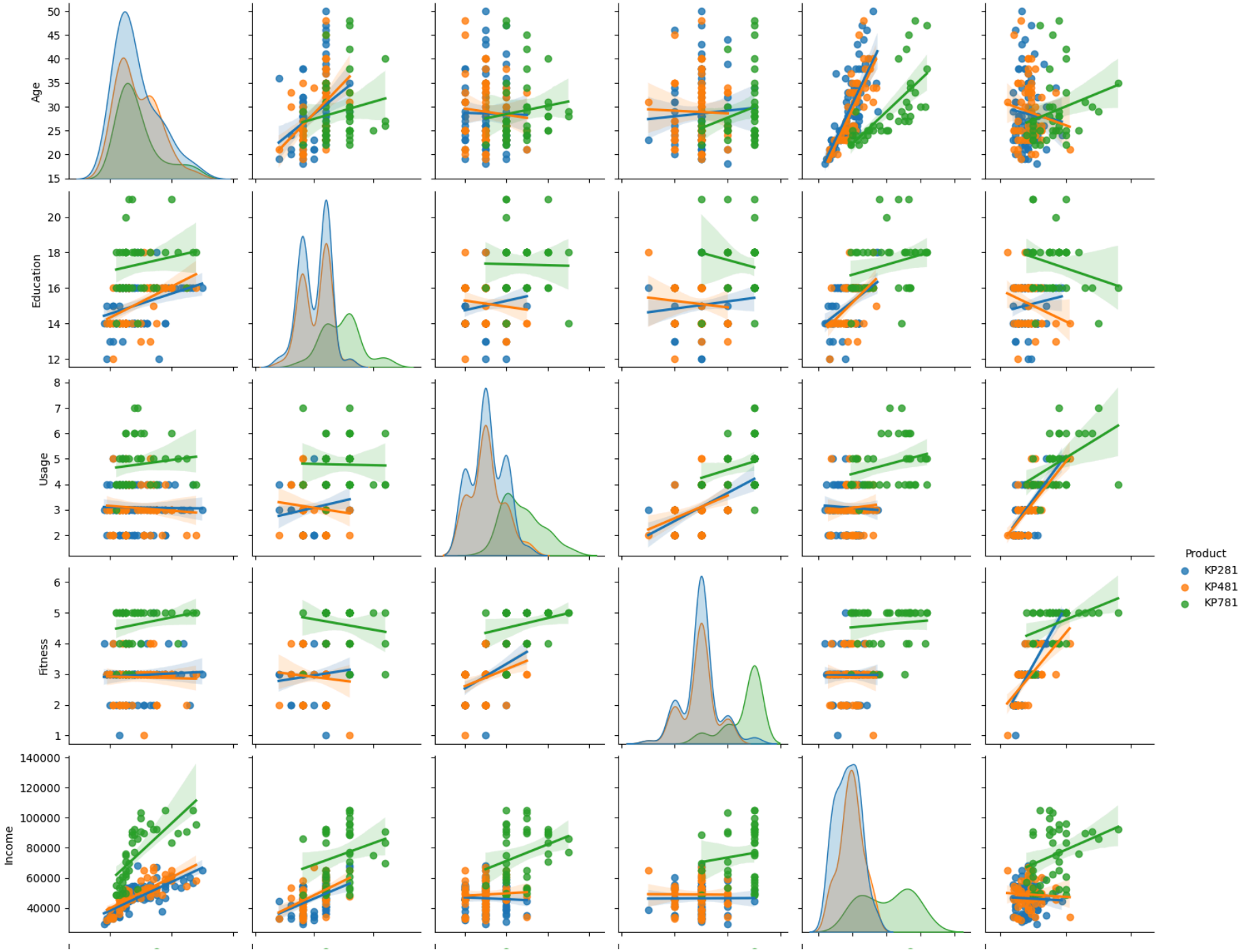
Out[ ]:

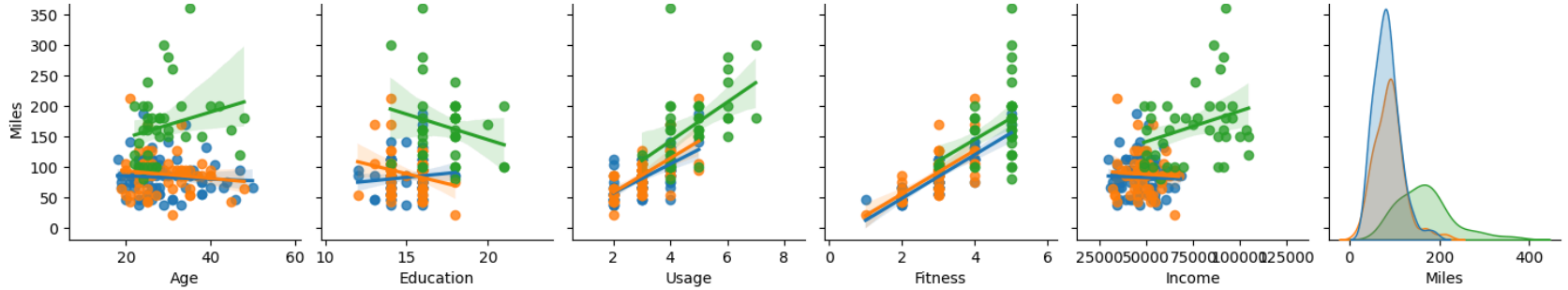
	Age	Education	Usage	Fitness	Income	Miles
Age	1.000000	0.280496	0.015064	0.061105	0.513414	0.036618
Education	0.280496	1.000000	0.395155	0.410581	0.625827	0.307284
Usage	0.015064	0.395155	1.000000	0.668606	0.519537	0.759130
Fitness	0.061105	0.410581	0.668606	1.000000	0.535005	0.785702
Income	0.513414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

In [ ]:
```
#Correlation HeatMap
plt.figure(figsize=(20,6))
ax = sns.heatmap(data_corr.corr(),annot=True,fmt='.4f',linewidths=.5,cmap='coolwarm')
plt.yticks(rotation=0)
plt.show()
```



In [ ]:
```
sns.pairplot(data, hue='Product',kind='reg')
plt.show()
```


Age
Education
Usage
Fitness
Income
Product
KP281
KP481
KP781



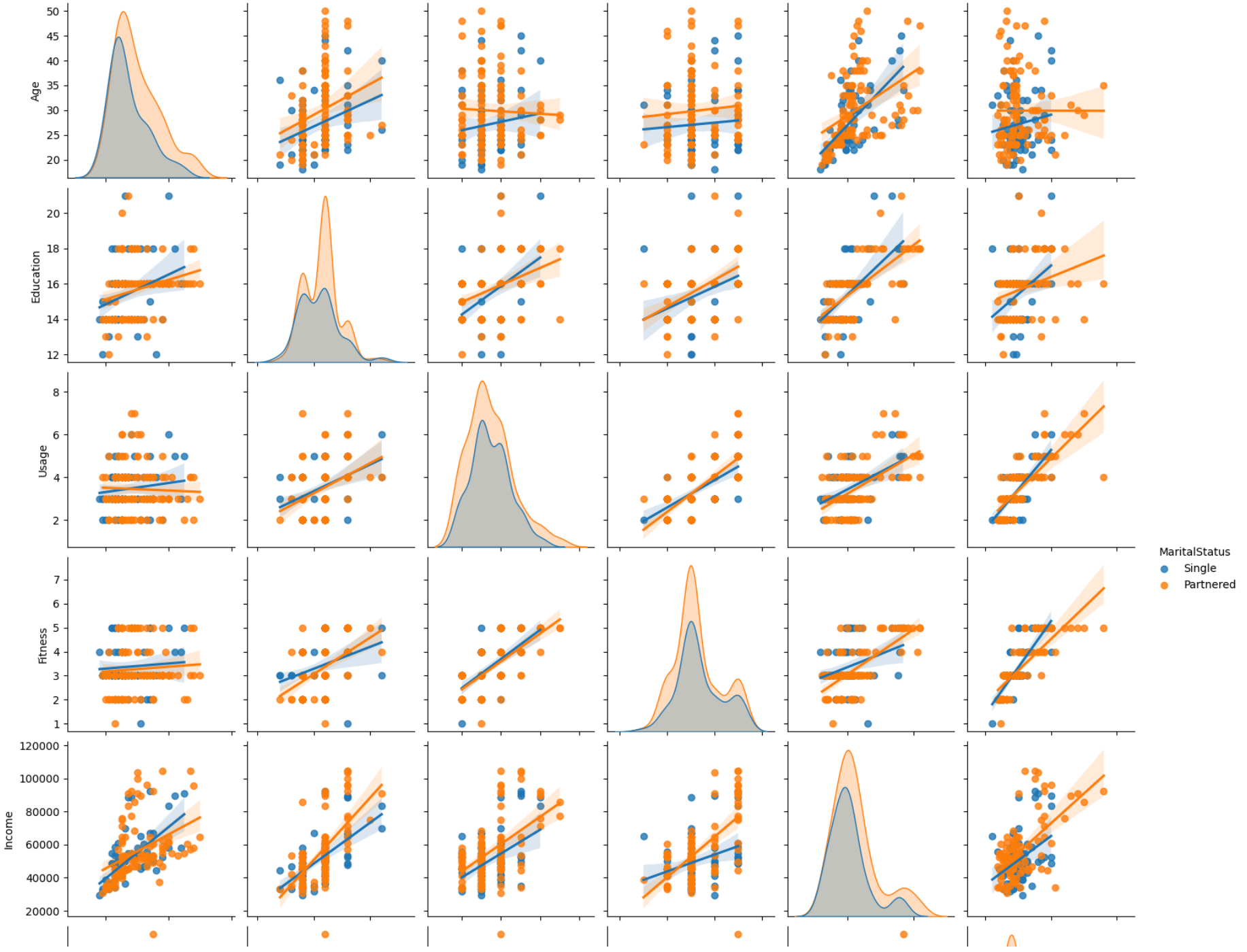
In [ ]:
```
data.head(1)
```

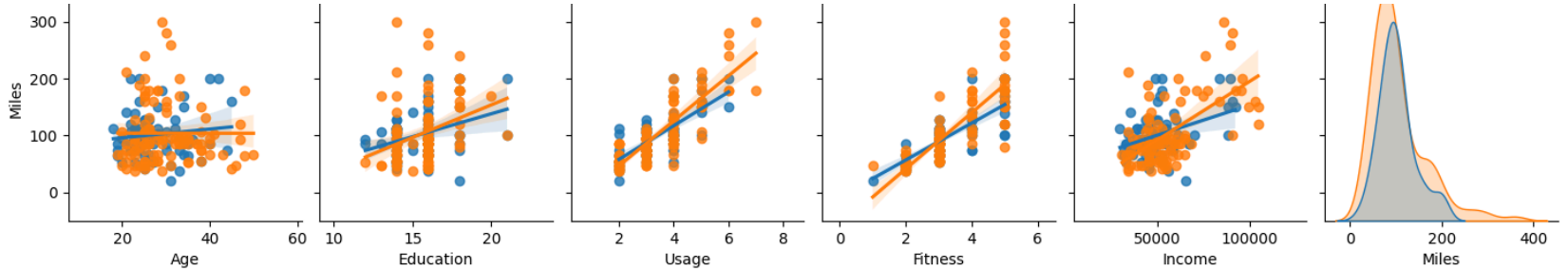
Out[ ]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles	Age_cat	Education_cat	income_cat	miles_cat	fitn
0	KP281	18	Male	14	Single	3	4	29562	112	Young_Adult	secondary	low	High_Activity	

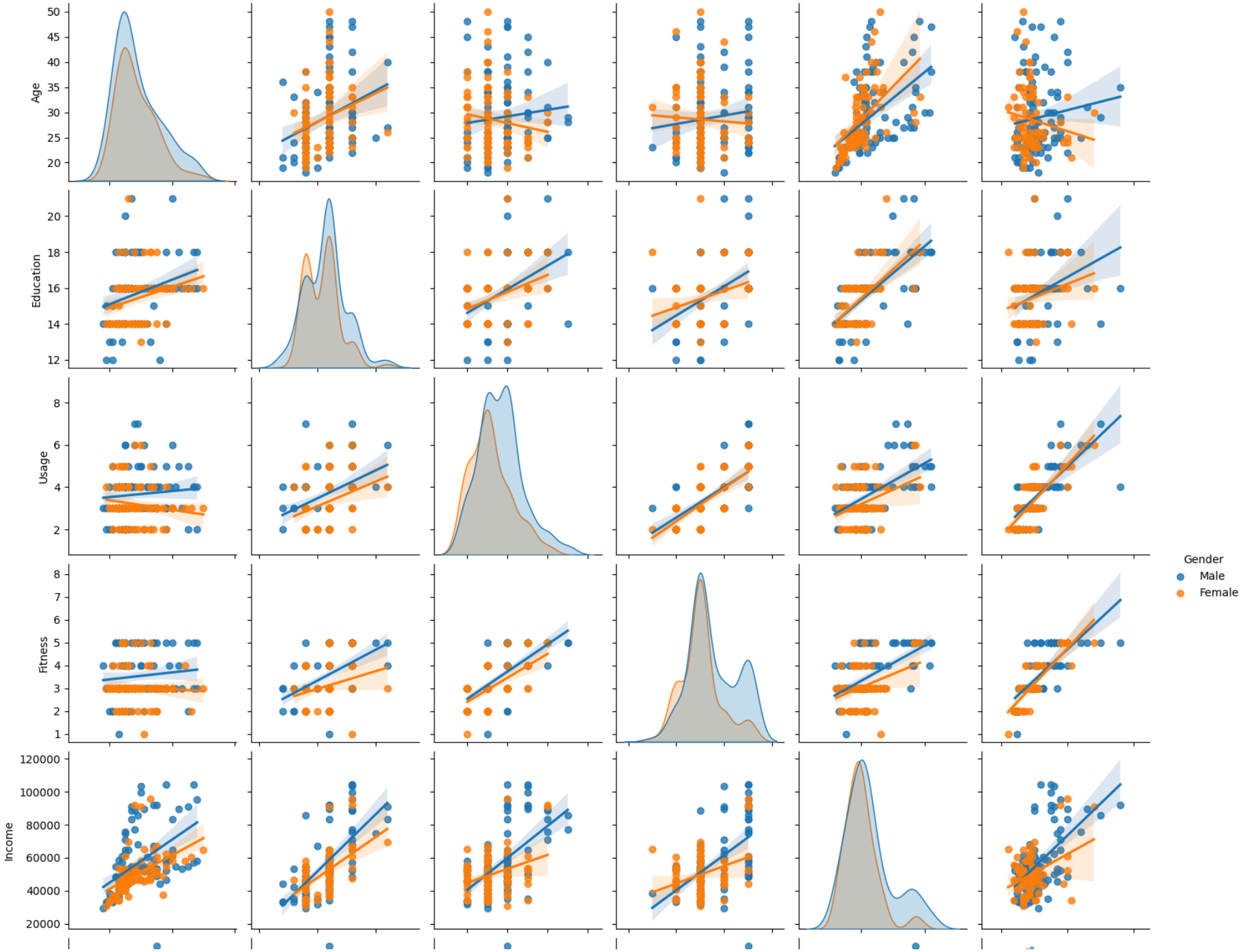
In [ ]:
```
plt.figure(figsize=(12, 10))
sns.pairplot(data, hue='MaritalStatus',kind='reg')
plt.show()
```

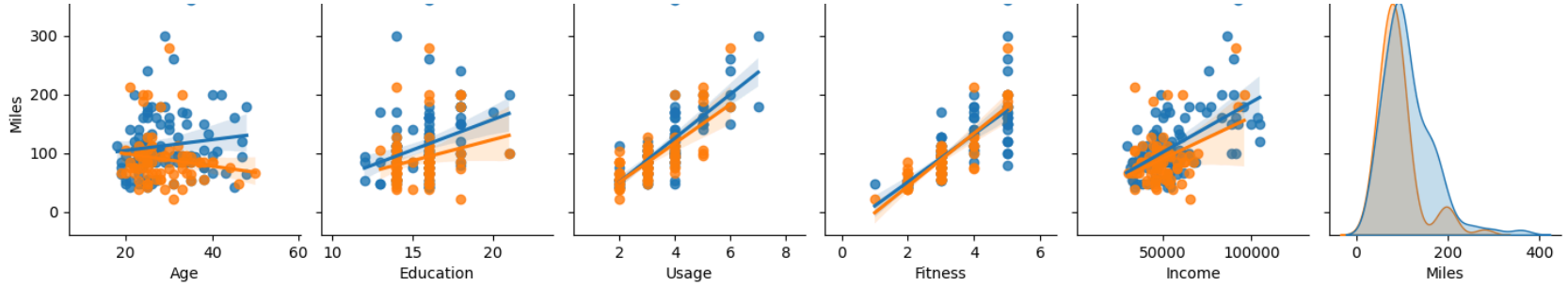
```
<Figure size 1200x1000 with 0 Axes>
```


Age
Education
Usage
Fitness
Income
MaritalStatus
Single
Partnered



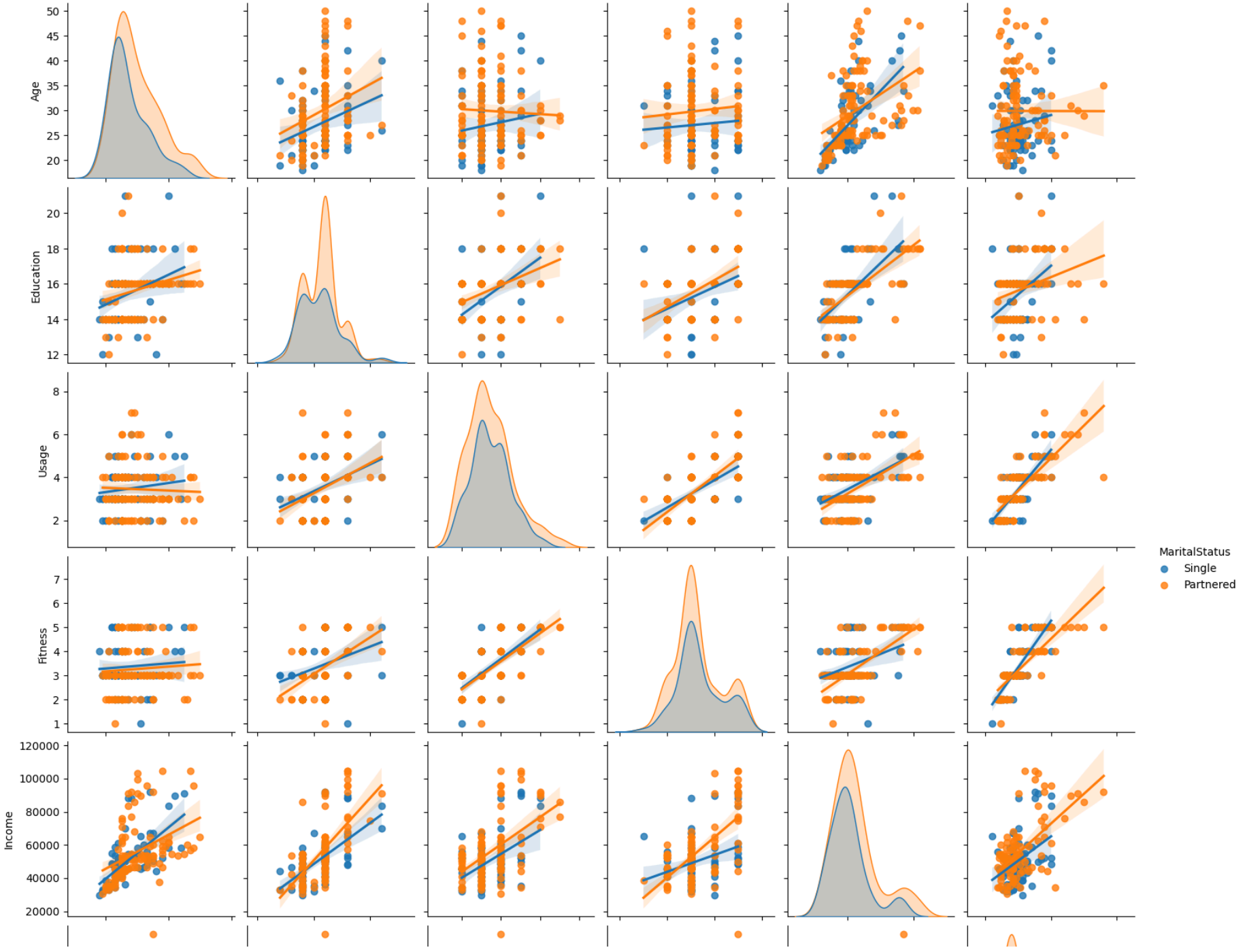
In [ ]:
```
sns.pairplot(data, hue='Gender',kind='reg')
plt.show()
```

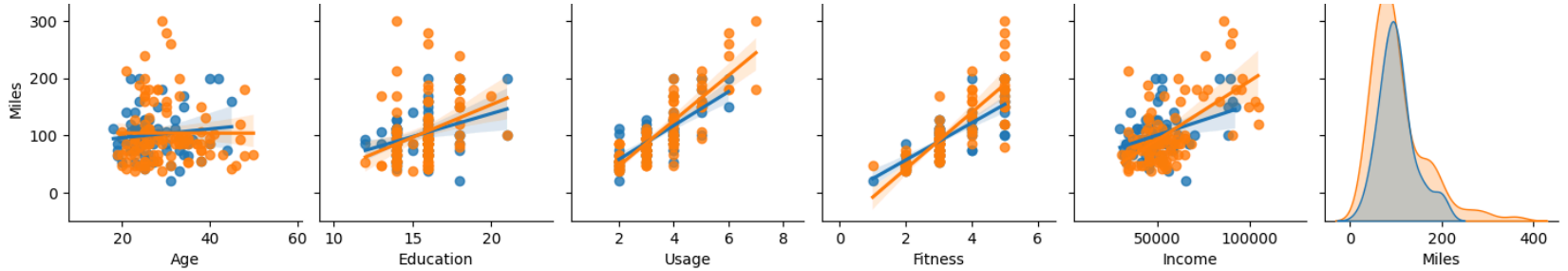

Age
Education
Usage
Fitness
Income
Gender
Male
Female



In [ ]:
```
sns.pairplot(data, hue='MaritalStatus',kind='reg')
```

Out[ ]:
```
<seaborn.axisgrid.PairGrid at 0x7de211779ab0>
```


Age
Education
Usage
Fitness
Income
MaritalStatus
Single
Partnered



Missing Value & Outlier Detection

In [ ]:
```
data.isnull().sum()
```

Out[ ]:

	0
Product	0
Age	0
Gender	0
Education	0
MaritalStatus	0
Usage	0
Fitness	0
Income	0
Miles	0
Age_cat	0
Education_cat	0

income_cat	0
miles_cat	0
fitness_cat	0

dtype: int64

In [ ]:
```
data.duplicated().sum()
```

Out[ ]: 0

In [ ]:
```
# Outlier calculation for Miles using Inter Quartile Range

q1, q3 = np.percentile(data.Miles, [25,75])
print(q1,",",  q3)

miles_iqr = q3 - q1

print("Inter Quartile Range Miles -->", miles_iqr)
```

```
66.0 , 114.75
Inter Quartile Range Miles --> 48.75
```

Business Insights based on Non-Graphical and Visual AnalysisNew Section

In [ ]:
```
data["Product"].value_counts(normalize=True).map(lambda x:round(x*100,2)).reset_index()
```

Out[ ]:

	Product	proportion
0	KP281	44.44
1	KP481	33.33
2	KP781	22.22

Among the three treadmill bought by cuatomers. the percentage as follows:-

1. **KP281 --> 44.44**
2. **KP481 --> 33.33**
3. **KP781 --> 22.22**

In [ ]:
```
data["fitness_cat"].value_counts(normalize=True).map(lambda x:round(x*100,2)).reset_index()
```

Out[ ]:

	fitness_cat	proportion
0	Average Shape	53.89
1	Excellent Shape	17.22
2	Bad Shape	14.44
3	Good Shape	13.33
4	Poor Shape	1.11

In [ ]:
```
data["Education_cat"].value_counts(normalize=True).map(lambda x:round(x*100,2)).reset_index()
```

Out[ ]:

	Education_cat	proportion
0	higher	62.22
1	secondary	36.11
2	primary	1.67

In [ ]:
```
data["Gender"].value_counts(normalize=True).map(lambda x:round(x*100,2)).reset_index()
```

Out[ ]:

	Gender	proportion

0	Male	57.78
1	Female	42.22

- There are two unique gender present in the dataset
- There are 57.78% Males and 42.22 are females.

In [ ]:
```
data["miles_cat"].value_counts(normalize=True).map(lambda x:round(x*100,2)).reset_index()
```

Out[ ]:

	miles_cat	proportion
0	Moderate_Activity	53.89
1	High_Activity	33.33
2	Light_Activity	9.44
3	Fitness Enthusiast	3.33

In [ ]:
```
data["MaritalStatus"].value_counts(normalize=True).map(lambda x:round(x*100,2)).reset_index()
```

Out[ ]:

	MaritalStatus	proportion
0	Partnered	59.44
1	Single	40.56

Probability for each product for the both Genders

In [ ]:
```
def gender_Probability(gender,data):
    print(f"Prob P(KP781) for {gender}: {round(data['KP781'][gender]/data.loc[gender].sum(),3)}")
    print(f"Prob P(KP481) for {gender}: {round(data['KP481'][gender]/data.loc[gender].sum(),3)}")
    print(f"Prob P(KP281) for {gender}: {round(data['KP281'][gender]/data.loc[gender].sum(),3)}")
```

```
df_temp = pd.crosstab(index=data['Gender'],columns=[data['Product']])
print("Prob of Male: ",round(df_temp.loc['Male'].sum()/len(data),3))
print("Prob of Female: ",round(df_temp.loc['Female'].sum()/len(data),3))
print()
gender_Probability('Male',df_temp)
print()
gender_Probability('Female',df_temp)
```

```
Prob of Male:  0.578
Prob of Female:  0.422

Prob P(KP781) for Male: 0.317
Prob P(KP481) for Male: 0.298
Prob P(KP281) for Male: 0.385

Prob P(KP781) for Female: 0.092
Prob P(KP481) for Female: 0.382
Prob P(KP281) for Female: 0.526
```

Probability of each product for given Marital Status

In [ ]:
```
def MS_Probability(ms_status,data):
    print(f"Prob P(KP781) for {ms_status}: {round(data['KP781'][ms_status]/data.loc[ms_status].sum(),3)}")
    print(f"Prob P(KP481) for {ms_status}: {round(data['KP481'][ms_status]/data.loc[ms_status].sum(),3)}")
    print(f"Prob P(KP281) for {ms_status}: {round(data['KP281'][ms_status]/data.loc[ms_status].sum(),3)}")

df_temp = pd.crosstab(index=data['MaritalStatus'],columns=[data['Product']])
print("Prob of P(Single): ",round(df_temp.loc['Single'].sum()/len(data),3))
print("Prob of P(Married/Partnered): ",round(df_temp.loc['Partnered'].sum()/len(data),3))
print()
MS_Probability('Single',df_temp)
print()
MS_Probability('Partnered',df_temp)
```

```
Prob of P(Single):  0.406
Prob of P(Married/Partnered):  0.594

Prob P(KP781) for Single: 0.233
Prob P(KP481) for Single: 0.329
Prob P(KP281) for Single: 0.438
```

```
Prob P(KP781) for Partnered: 0.215
Prob P(KP481) for Partnered: 0.336
Prob P(KP281) for Partnered: 0.449
```

In [97]:
```
data.Product.value_counts(normalize=True)
```

Out[97]:

Product	proportion
KP281	0.444444
KP481	0.333333
KP781	0.222222

dtype: float64

In [123…
```
# Conditional and Marginal Probabilities with product type and age group
pd.crosstab(index=data.Product, columns=data.Age_cat, margins=True)
```

Out[123…

Product	Age_cat	Young_Adult	Adults	Middle_Aged_Adults	Elder	All
KP281		34	32	11	3	80
KP481		28	24	7	1	60
KP781		17	17	4	2	40
All		79	73	22	6	180

In [111…
```
pd.crosstab(index=data.Product, columns=data.Age_cat, margins=True, normalize="columns")*100
```

Out [111…

Age_cat	Young_Adult	Adults	Middle_Aged_Adults	Elder	All
Product					
KP281	43.037975	43.835616	50.000000	50.000000	44.444444
KP481	35.443038	32.876712	31.818182	16.666667	33.333333
KP781	21.518987	23.287671	18.181818	33.333333	22.222222

In [118…

```
(pd.crosstab(index=data.Product, columns=data.Age_cat, margins=True, normalize=True)*100).round(2)
```

Out [118…

Age_cat	Young_Adult	Adults	Middle_Aged_Adults	Elder	All
Product					
KP281	18.89	17.78	6.11	1.67	44.44
KP481	15.56	13.33	3.89	0.56	33.33
KP781	9.44	9.44	2.22	1.11	22.22
All	43.89	40.56	12.22	3.33	100.00

In [125…

```
# Conditional and Marginal Probabilities with product type and Fitness category
(pd.crosstab(index=data.Product, columns=data.fitness_cat, margins=True)*100).round(2)
```

Out [125…

fitness_cat	Average Shape	Bad Shape	Excellent Shape	Good Shape	Poor Shape	All
Product						
KP281	5400	1400	200	900	100	8000
KP481	3900	1200	0	800	100	6000
KP781	400	0	2900	700	0	4000

All	9700	2600	3100	2400	200	18000

In [127…
```
pd.crosstab(index=[data.Product,data.fitness_cat], columns=data.Gender)
```

Out[127…

Product	fitness_cat	Gender Female	Male
KP281	Average Shape	26	28
	Bad Shape	10	4
	Excellent Shape	1	1
	Good Shape	3	6
	Poor Shape	0	1
KP481	Average Shape	18	21
	Bad Shape	6	6
	Good Shape	4	4
	Poor Shape	1	0
KP781	Average Shape	1	3
	Excellent Shape	5	24
	Good Shape	1	6

In [130…
```
(pd.crosstab(index=[data.Product,data.fitness_cat], columns=data.Gender, normalize=True)*100).round(2)
```

Out[130…

Product	fitness_cat	Gender Female	Male

Product	fitness_cat		
KP281	Average Shape	14.44	15.56
	Bad Shape	5.56	2.22
	Excellent Shape	0.56	0.56
	Good Shape	1.67	3.33
	Poor Shape	0.00	0.56
KP481	Average Shape	10.00	11.67
	Bad Shape	3.33	3.33
	Good Shape	2.22	2.22
	Poor Shape	0.56	0.00
KP781	Average Shape	0.56	1.67
	Excellent Shape	2.78	13.33
	Good Shape	0.56	3.33

In [128…
```
pd.crosstab(index=[data.Product,data.fitness_cat], columns=data.MaritalStatus)
```

Out[128…

	MaritalStatus	Partnered	Single
Product	fitness_cat		
KP281	Average Shape	31	23
	Bad Shape	11	3
	Excellent Shape	1	1

	Good Shape	4	5
	Poor Shape	1	0
KP481	Average Shape	25	14
	Bad Shape	7	5
	Good Shape	4	4
	Poor Shape	0	1
KP781	Average Shape	1	3
	Excellent Shape	17	12
	Good Shape	5	2

In [131…
```
(pd.crosstab(index=[data.Product,data.fitness_cat], columns=data.MaritalStatus, normalize=True)*100).round(2)
```

Out[131…

	MaritalStatus	Partnered	Single
Product	fitness_cat		
KP281	Average Shape	17.22	12.78
	Bad Shape	6.11	1.67
	Excellent Shape	0.56	0.56
	Good Shape	2.22	2.78
	Poor Shape	0.56	0.00
KP481	Average Shape	13.89	7.78
	Bad Shape	3.89	2.78

	Good Shape	2.22	2.22
	Poor Shape	0.00	0.56
KP781	Average Shape	0.56	1.67
	Excellent Shape	9.44	6.67
	Good Shape	2.78	1.11

In [132…
```
pd.crosstab(index=[data.Product,data.MaritalStatus], columns=data.Gender)
```

Out[132…

	Gender	Female	Male
Product	MaritalStatus		
KP281	Partnered	27	21
	Single	13	19
KP481	Partnered	15	21
	Single	14	10
KP781	Partnered	4	19
	Single	3	14

In [136…
```
(pd.crosstab(index=[data.Product,data.MaritalStatus], columns=data.Gender, normalize=True)*100).round(2)
```

Out[136…

	Gender	Female	Male
Product	MaritalStatus		
KP281	Partnered	15.00	11.67

	Single	7.22	10.56
KP481	Partnered	8.33	11.67
	Single	7.78	5.56
KP781	Partnered	2.22	10.56
	Single	1.67	7.78

In [144…
```
(pd.crosstab(index=data.Product, columns=data.Gender,normalize= "columns", margins=True)*100).round(2)
```

Out[144…

Gender	Female	Male	All
Product			
KP281	52.63	38.46	44.44
KP481	38.16	29.81	33.33
KP781	9.21	31.73	22.22

Probability of Selling Product

- The KP281 Products is brought by Females is 52%
- The KP481 Products is brought by Females is 38%
- The KP781 Products is brought by Females is 9%
- The KP281 Products is brought by Males is 38%
- The KP481 Products is brought by Males is 29%
- The KP781 Products is brought by Males is 31%

- The KP281 product is brought by most of the customers 44%
- The KP481 Products is brought by less coustomers compared to KP281 i.e, 33%
- The KP781 Products low selling product with 22%

The pricing of the products having the effects of the purchese of the produts:-

- 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 [146…
```
(pd.crosstab(index=data.Product, columns=data.income_cat,normalize= "columns", margins=True)*100).round(2)
```

Out[146…

income_cat	low	Moderate	High	very_high	All
Product					
KP281	71.88	48.11	26.09	0.0	44.44
KP481	28.12	41.51	30.43	0.0	33.33
KP781	0.00	10.38	43.48	100.0	22.22

In [151…
```
(pd.crosstab(index=data.Product, columns=data.fitness_cat,normalize= "columns", margins=True)*100).round(2)
```

Out[151…

fitness_cat	Average Shape	Bad Shape	Excellent Shape	Good Shape	Poor Shape	All
Product						
KP281	55.67	53.85	6.45	37.50	50.0	44.44
KP481	40.21	46.15	0.00	33.33	50.0	33.33

KP781	4.12	0.00	93.55	29.17	0.0	22.22

In [152…
```
(pd.crosstab(index=data.Product, columns=data.Age_cat,normalize= "columns", margins=True)*100).round(2)
```

Out[152…

Age_cat	Young_Adult	Adults	Middle_Aged_Adults	Elder	All
Product					
KP281	43.04	43.84	50.00	50.00	44.44
KP481	35.44	32.88	31.82	16.67	33.33
KP781	21.52	23.29	18.18	33.33	22.22

In [153…
```
(pd.crosstab(index=data.Product, columns=data.Education_cat,normalize= "columns", margins=True)*100).round(2)
```

Out[153…

Education_cat	primary	secondary	higher	All
Product				
KP281	66.67	56.92	36.61	44.44
KP481	33.33	40.00	29.46	33.33
KP781	0.00	3.08	33.93	22.22

In [154…
```
(pd.crosstab(index=data.Product, columns=data.Usage,normalize= "columns", margins=True)*100).round(2)
```

Out[154…

Usage	2	3	4	5	6	7	All
Product							
KP281	57.58	53.62	42.31	11.76	0.0	0.0	44.44

KP481	42.42	44.93	23.08	17.65	0.0	0.0	33.33
KP781	0.00	1.45	34.62	70.59	100.0	100.0	22.22

Customer Profiling for Each Product

1. KP281:-

- This is the entry level treadmill that sells for $1,500
- Most of the customers brought this product falling under Low income category.
- The Young adults, Adults, Middle age Aduluts are mostly used
- People using this product having Average Shape
- The Primary & Secrondary education people are more.
- Partner Famales having product compare single.
- Overall 44% of customers brought this product.
- This product is easily afforded by both Male and Female customers.
- More people are doing light activity.
- Most of the poeple falls under low inocme (Upto 40,000 Dollers)

---

2. KP481:-

- This is intermediate level brought
- The cost of product is selling about $1,750.
- About 33% of customers having this product.
- Most brought product as compared to KP281.
- Second most used product.
- Customers having education Primary & Secondary brought most.
- The customers having average shape is about 40% &good shape are 33%
- More females are prefering this product than males
- They are doing moderate activity.
- Most of the poeple falls under moderate income (40,000 to 60,000 Dollers)

*The more Females brought this product than Males

3. KP781 :-

- This product is having advance features compare with all
- The cost of prodct is $2,500
- About 22% of people brought this product
- Most of the users are fitness Enthusiast
- The customers are pursed higher education
- Most of elders having this product.
- The more males brought this product than Females
- More people are doing high activity.
- The users are of Excellent Shape of body.
- The high & very high income gorup people brought this product

Recommendation

- Female who prefer exercising equipments are very low here. Hence, we should run a marketing campaign on to encourage women to exercise more
- KP281 & KP481 treadmills are preferred by the customers whose annual income lies in the range of 39K - 53K Dollars. These models should promoted as budget treadmills.
- As KP781 provides more features and functionalities, the treadmill should be marketed for professionals and athletes.
- KP781 product should be promotted using influencers and other international atheletes.
- KP781 can be recommended for Female customers who exercises extensively along with easy usage guidance since this type is advanced.