Web search Copy

Double-click (or enter) to edit

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.

Features of the dataset: -Feature Description

- Product Product Purchased: KP281, KP481, or KP781
- Age Age of buyer in years
- Gender Gender of buyer (Male/Female)
- Education Education of buyer in years
- MaritalStatus MaritalStatus of buyer (Single or partnered)
- Usage The average number of times the buyer plans to use the - treadmill each week
- Income Annual income of the buyer (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 buyer expects to walk/run each week

Double-click (or enter) to edit

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
aerofit = pd.read_csv('aerofit_treadmill.csv')

df=aerofit.copy()
df.head()
```

	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

Next steps: Generate code with df View recommended plots New interactive sheet

Generate Using dataframe: df provide code for all above plots Close

1 of 1 Use code with caution

```
# prompt: Using dataframe df: provide code for all above plots

# Import necessary libraries
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming your DataFrame is named 'df'


# Plot 1: Histogram of Age
plt.figure(figsize=(8, 6))
sns.histplot(df['Age'], bins=10, kde=True)
plt.title('Distribution of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```

Web search Copy

```
Income by Gender
, 6))
sns.boxplot(x='Gender', y='Income', data=df)
plt.title('Income Distribution by Gender')
plt.xlabel('Gender')
plt.ylabel('Income')
plt.show()


# Plot 3: Scatter plot of Miles vs. Usage
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Miles', y='Usage', data=df)
plt.title('Miles vs. Usage')
plt.xlabel('Miles')
plt.ylabel('Usage')
plt.show()


# Plot 4: Countplot of Product
plt.figure(figsize=(8, 6))
sns.countplot(x='Product', data=df)
plt.title('Count of Products')
plt.xlabel('Product')
plt.ylabel('Count')
plt.show()


# Plot 5: Violin plot of Education by MaritalStatus
plt.figure(figsize=(8, 6))
sns.violinplot(x='MaritalStatus', y='Education', data=df)
plt.title('Education Distribution by Marital Status')
plt.xlabel('Marital Status')
plt.ylabel('Education')
plt.show()


# Plot 6: Pairplot of selected variables
sns.pairplot(df[['Age', 'Income', 'Miles', 'Usage']])
plt.show()


# Plot 7: Heatmap of correlation between variables
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()



df.info()
```

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

```
df.shape
```

```
(180, 9)
```

Changing the Datatype of Columns

Web search Copy

```
    'object':
    df[col] = df[col].astype('category')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180 entries, 0 to 179
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Product        180 non-null    category
 1   Age            180 non-null    int64
 2   Gender         180 non-null    category
 3   Education      180 non-null    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: 9.5 KB
```

Statistical summary

```
df.describe()
```

	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

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Age	180.0	28.788889	6.943498	18.0	24.00	26.0	33.00	50.0
Education	180.0	15.572222	1.617055	12.0	14.00	16.0	16.00	21.0
Usage	180.0	3.455556	1.084797	2.0	3.00	3.0	4.00	7.0
Fitness	180.0	3.311111	0.958869	1.0	3.00	3.0	4.00	5.0
Income	180.0	53719.577778	16506.684226	29562.0	44058.75	50596.5	58668.00	104581.0
Miles	180.0	103.194444	51.863605	21.0	66.00	94.0	114.75	360.0

data insights:

1. Age - The age range of customers spans from 18 to 50 year, with an average age of 29 years.
2. Education - Customer education levels vary between 12 and 21 years, with an average education duration of 16 years.
3. Usage - Customers intend to utilize the product anywhere from 2 to 7 times per week, with an average usage frequency of 3 times per week.
4. Fitness - On average, customers have rated their fitness at 3 on a 5-point scale, reflecting a moderate level of fitness.
5. Income - The annual income of customers falls within the range of USD 30,000 to USD 100,000, with an average income of approximately USD 54,000.
6. Miles - Customers' weekly running goals range from 21 to 360 miles, with an average target of 103 miles per week.

Web search Copy

```
category').T
```

	count	unique	top	freq
Product	180	3	KP281	80
Gender	180	2	Male	104
MaritalStatus	180	2	Partnered	107

Category data insights:

1. Product - Over the past three months, the KP281 product demonstrated the highest sales performance among the three products
2. Gender - Based on the data of last 3 months, around 58% of the buyers were Male and 42% were female
3. Marital Status - Based on the data of last 3 months, around 60% of the buyers were Married and 40% were single

```
# duplicates
df.duplicated().sum()
```

```
0
```

```
df[df.duplicated()]
```

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles

There are no duplicate entries in the dataset

Handling Missing values

```
df_missing = df.isna().sum()
df_missing
```

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

dtype: int64

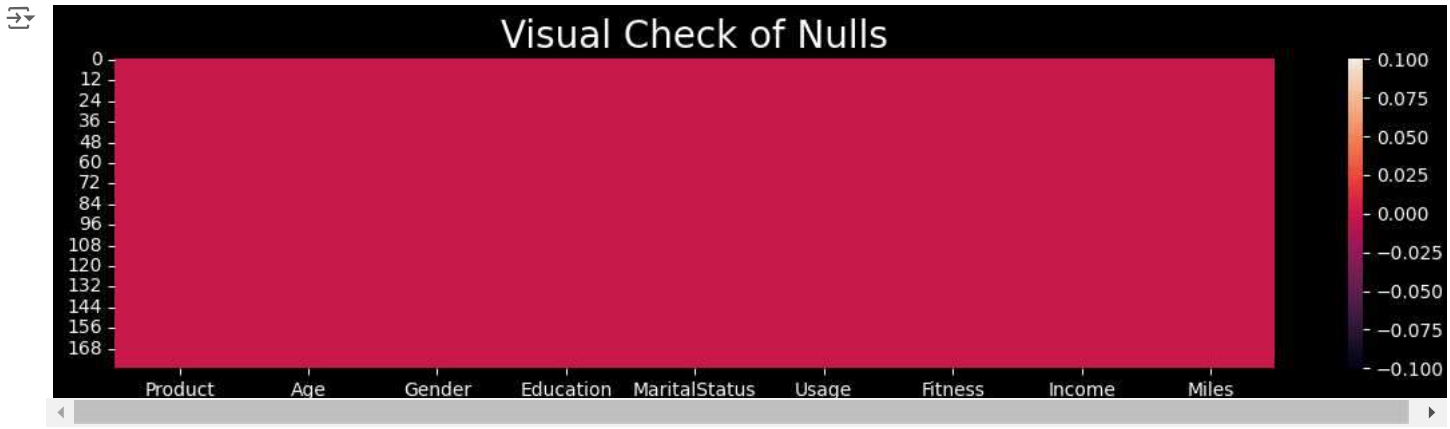
```
df.isna().any()
```

Web search | Copy

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

dtype: bool

```
plt.figure(figsize=(14,3))
plt.style.use('dark_background')
sns.heatmap(df.isnull())
plt.title('Visual Check of Nulls',fontsize=20)
plt.show()
```



There are no missing entries in the dataset

```
print('Parameter Ranges:')
print(f"Age : {df.Age.min()}yrs - {df.Age.max()}yrs")
print(f"Education : {df.Education.min()} - {df.Education.max()} yrs")
print(f"Usage : {df.Usage.min()} - {df.Usage.max()} days per week")
print(f"Fitness : {df.Fitness.min()} - {df.Fitness.max()} in a scale")
print(f"Income : {round(df.Income.min()//1000)}k - {round(df.Income.max()//1000)}k in $")
print(f"Miles : {df.Miles.min()} - {df.Miles.max()} miles per week")
```

```
Parameter Ranges:
Age : 18yrs - 50yrs
Education : 12 - 21 yrs
Usage : 2 - 7 days per week
Fitness : 1 - 5 in a scale
Income : 29k - 104k in $
Miles : 21 - 360 miles per week
```

```
#checking the unique values for columns
for col in df.columns:
    print()
    print('Total Unique Values in',col,'column are :-',df[col].nunique())
    print('Unique Values in',col,'column are :-\n',df[col].unique())
    print()
    print('-'*100)
```

```
Total Unique Values in Product column are :- 3
Unique Values in Product column are :-
 ['KP281', 'KP481', 'KP781']
Categories (3, object): ['KP281', 'KP481', 'KP781']

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

```
ues in Age column are :- 32
Unique Values in Age column are :-
 [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]

--------------------------------------------------------------------------------------------------------

Total Unique Values in Gender column are :- 2
Unique Values in Gender column are :-
 ['Male', 'Female']
Categories (2, object): ['Female', 'Male']

--------------------------------------------------------------------------------------------------------

Total Unique Values in Education column are :- 8
Unique Values in Education column are :-
 [14 15 12 13 16 18 20 21]

--------------------------------------------------------------------------------------------------------

Total Unique Values in MaritalStatus column are :- 2
Unique Values in MaritalStatus column are :-
 ['Single', 'Partnered']
Categories (2, object): ['Partnered', 'Single']

--------------------------------------------------------------------------------------------------------

Total Unique Values in Usage column are :- 6
Unique Values in Usage column are :-
 [3 2 4 5 6 7]

--------------------------------------------------------------------------------------------------------

Total Unique Values in Fitness column are :- 5
Unique Values in Fitness column are :-
 [4 3 2 1 5]

--------------------------------------------------------------------------------------------------------

Total Unique Values in Income column are :- 62
Unique Values in Income column are :-
 [ 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]

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

```
for col in df.columns:
    if df[col].dtype != 'category':
        print(f'Value_counts of {col} are :- \n{df[col].value_counts().to_frame().reset_index()}')
        print()
        print('-'*100)
```

Web search | Copy

```
8      53      7
9     100      7
10    180      6
11    200      6
12     56      6
13     64      6
14    127      5
15    160      5
16     42      4
17    150      4
18     38      3
19     74      3
20    170      3
21    120      3
22    103      3
23    132      2
24    141      2
25    280      1
26    260      1
27    300      1
28    240      1
29    112      1
30    212      1
31     80      1
32    140      1
33     21      1
34    169      1
35    188      1
36    360      1
```

- There a 180 rows and 9 columns with most numerical data.
- There are no missing values in the data.
- There are 3 unique products ('KP281', 'KP481', 'KP781').
- KP281 is the most frequent product.
- Males are generally the frequent customers.
- The standard deviation for Income & Miles is very high so there might be a possibility of outliers in those attibutes.
- The Average age of the customers is 28 and they have approx 16 yrs of education.
- The Mean Usage per week is 3.4, with maximum as 7 and minimum as 2.
- The Average fitness rating is 3.3 on a scale of 1 to 5.
- The Average number of miles the customer walks is 103 with max is almost 115 and minimum of 21.
- There are 107 partnered customers and 73 single customers.
- On an Average , customers use threadmill THRICE a week.

```
df.sample()
```

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
77	KP281	46	Female	16	Partnered	3	2	60261	47

```
df.Product.value_counts()
```

Product	count
KP281	80
KP481	60
KP781	40

dtype: int64

```
fp = df.Product.value_counts(normalize=True).to_frame()
fp = fp.reset_index()
fp['proportion'] = round(fp.proportion*100,2)
fp
```

Web search | Copy

		portion
0	KP281	44.44
1	KP481	33.33
2	KP781	22.22

Next steps: Generate code with fp | View recommended plots | New interactive sheet

KP281 has highest sales of 44%

- 44.44% of customers bought KP281 product type
- 33.33% of customers bought KP481 product type
- 22.22% of customers bought KP781 product type

Almost 85% of the customers plan to use the treadmill for 2 to 4 times a week and only 15% using 5 times and above each week

54% of the customers have self-evaluated their fitness at a level 3 on a scale of 1 to 5. Furthermore, a substantial 84% of the total customers have rated themselves at 3 or higher, indicating commendable fitness levels.

```
# Customer Gender statistics (listed in %)
fg = df['Gender'].value_counts(normalize=True)
fg = fg.map(lambda z: round(z*100,2))
fg
```

Gender	proportion
Male	57.78
Female	42.22

dtype: float64

57.78% of customers are Male

42.22% customers are Female

```
# Customer Marital Status statistics (listed in %)
mf= df['MaritalStatus'].value_counts(normalize=True)
mf = mf.map(lambda z:round(z*100,2))
mf
```

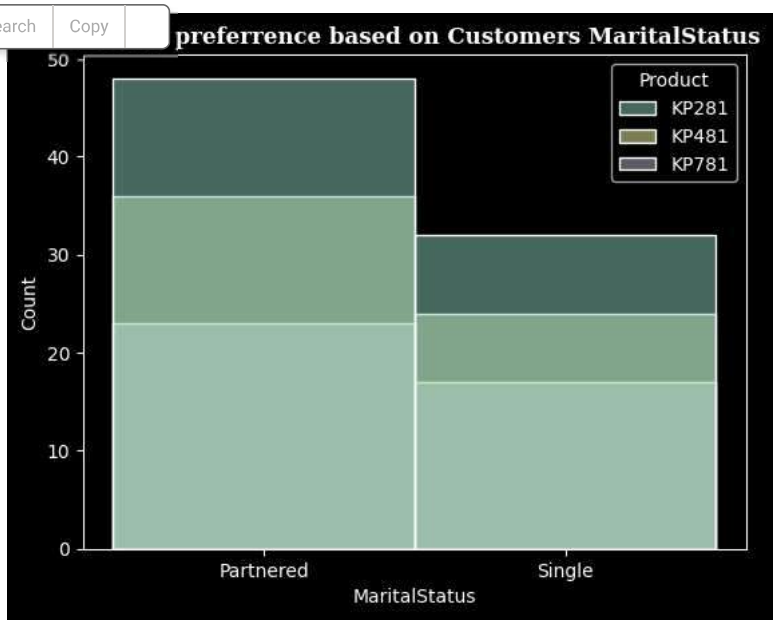
MaritalStatus	proportion
Partnered	59.44
Single	40.56

dtype: float64

- 59.44% of customers are partnered
- 40.56% customers are single

```
sns.histplot(data=df, x="MaritalStatus", hue="Product" )
plt.title('Product preferrence based on Customers MaritalStatus',fontfamily='serif',fontweight='bold',fontsize=12,color='w')
plt.show()
```

Web search Copy




```
# Customers Usage - Number of days used per week (listed in %)
cu = df['Usage'].value_counts(normalize=True).map(lambda z:round(z*100,2)).reset_index()
cu = cu.rename({'Usage':'DaysPerWeek'},axis=1)
cu
```

	DaysPerWeek	proportion
0	3	38.33
1	4	28.89
2	2	18.33
3	5	9.44
4	6	3.89
5	7	1.11

Next steps: Generate code with cu | View recommended plots | New interactive sheet

- Around 38% of customers use 3 days per week
- Less than 4% of customers use 6 days per week

```
# Categorization of Fitness Rating where 1 is the poor shape and 5 is the excellent shape.

df['Fitness_comment'] = df.Fitness
df['Fitness_comment'] = df.Fitness_comment.replace({
                            1:"Poor Shape",
                            2:"Bad Shape",
                            3:"Average Shape",
                            4:"Good Shape",
                            5:"Excellent Shape"})
df.tail()
```

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles	age_category	Fitness_comment
175	KP781	40	Male	21	Single	6	5	83416	200	Middle Aged	Excellent Shape
176	KP781	42	Male	18	Single	5	4	89641	200	Middle Aged	Good Shape
177	KP781	45	Male	16	Single	5	5	90886	160	Middle Aged	Excellent Shape
178	KP781	47	Male	18	Partnered	4	5	104581	120	Elderly	Excellent Shape
179	KP781	48	Male	18	Partnered	4	5	95508	180	Elderly	Excellent Shape

```
# Customers Fitness - Number of days used per week (listed in %)
ffc = df['Fitness_comment'].value_counts(normalize=True)
```

Web search Copy

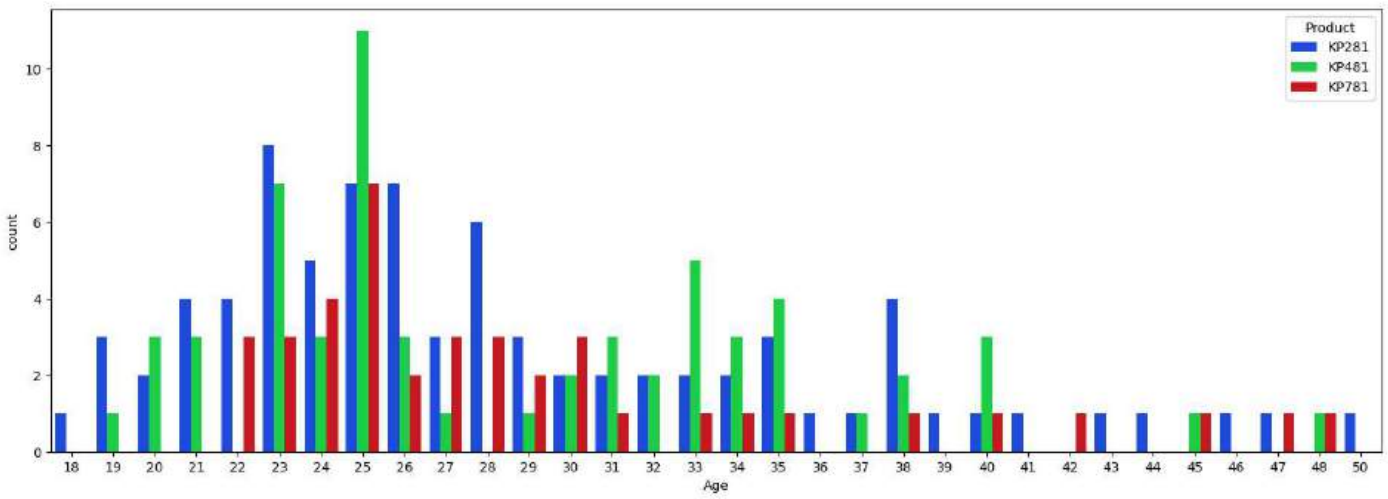
```
z:round(z*100,2)).reset_index()
hape','Bad Shape','Average Shape','Good Shape','Excellent Shape']
ffc['Fitness_comment'] = pd.Categorical(ffc['Fitness_comment'],categories=manual_order, ordered=True)
ffc = ffc.sort_values(by='Fitness_comment').reset_index(drop=True)
ffc
```

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

Next steps: Generate code with ffc | View recommended plots | New interactive sheet

Approx 54% of customers have rated themselves as they are in Average Shape Little close to 14% of customers have rated their fitness less than average Over 17% of customers have Peak fitness ratings

```
plt.figure(figsize = (18,6))
plt.style.use('default')
plt.style.use('seaborn-v0_8-bright')
sns.countplot(data=df, x='Age',hue="Product")
plt.suptitle('Products Preferences based on Customers Age',fontfamily='serif',fontweight='bold',fontsize=20,color='w')

plt.show()
```




```
# Categorization of age:

# 0-21 -> Teenage
# 22-35 -> Adults
# 36-45 -> Middle Age
# 46-60 -> Elderly person

df['age_category'] = df.Age
df['age_category'] = pd.cut(df.age_category,bins=[0,20,35,45,60],labels=['Teenage','Adults','Middle Aged','Elderly'])
df.sample(2)
```

Web search Copy

		ge	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles	age_category
25	KP281	24	Male	13	Partnered	3	2	42069	47	Adults
138	KP481	45	Male	16	Partnered	2	2	54576	42	Middle Aged

```
df.age_category.value_counts()
```

age_category	count
Adults	142
Middle Aged	22
Teenage	10
Elderly	6

dtype: int64

```
acc = df.groupby(['Product','age_category']).agg(cnt=('Product','count'))
acc.reset_index(inplace=True)
acc
```

```
<ipython-input-56-2661f4baa344>:1: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future ve
  acc = df.groupby(['Product','age_category']).agg(cnt=('Product','count'))
```

	Product	age_category	cnt
0	KP281	Teenage	6
1	KP281	Adults	60
2	KP281	Middle Aged	11
3	KP281	Elderly	3
4	KP481	Teenage	4
5	KP481	Adults	48
6	KP481	Middle Aged	7
7	KP481	Elderly	1
8	KP781	Teenage	0
9	KP781	Adults	34
10	KP781	Middle Aged	4
11	KP781	Elderly	2

Next steps: Generate code with acc View recommended plots New interactive sheet

```
# Categorization of income:

df.Income.describe()
```

	Income
count	180.000000
mean	53719.577778
std	16506.684226
min	29562.000000
25%	44058.750000
50%	50596.500000
75%	58668.000000
max	104581.000000

dtype: float64

Web search Copy

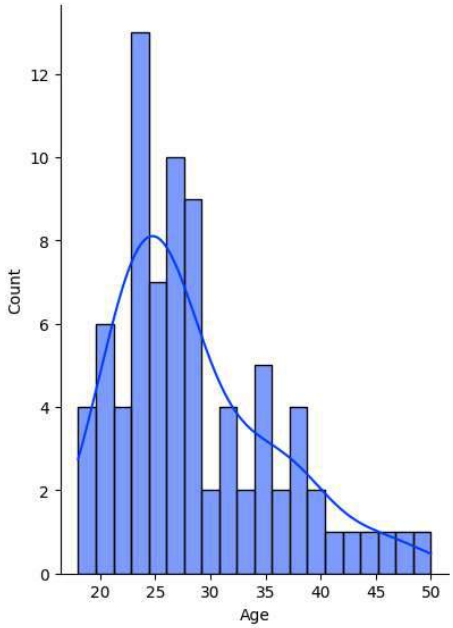
```
.cut(df.Income , bins=[25000,50000,75000,150000] , labels=['middle_class','upper_middle_class','high_class'])
```

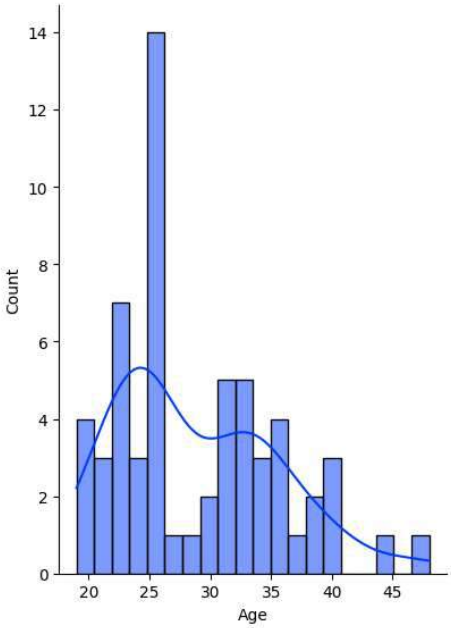
```
df.sample(4)
```

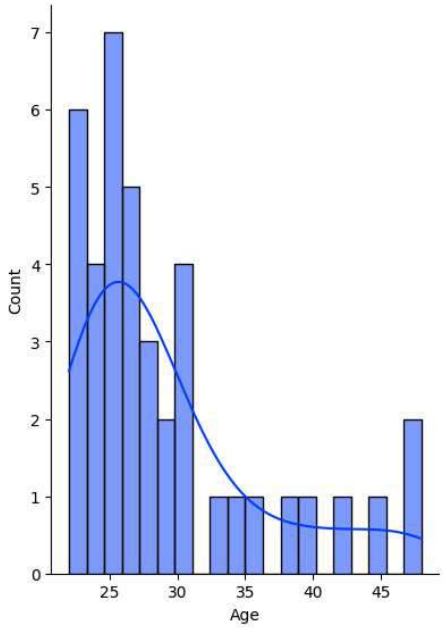
	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles	age_category	Fitness_comment	income_grp
83	KP481	20	Male	14	Single	3	3	38658	95	Teenage	Average Shape	middle_class
46	KP281	28	Male	14	Single	3	3	52302	103	Adults	Average Shape	upper_middle_class
64	KP281	35	Female	16	Partnered	3	3	60261	94	Adults	Average Shape	upper_middle_class
58	KP281	32	Male	14	Partnered	4	3	52302	85	Adults	Average Shape	upper_middle_class

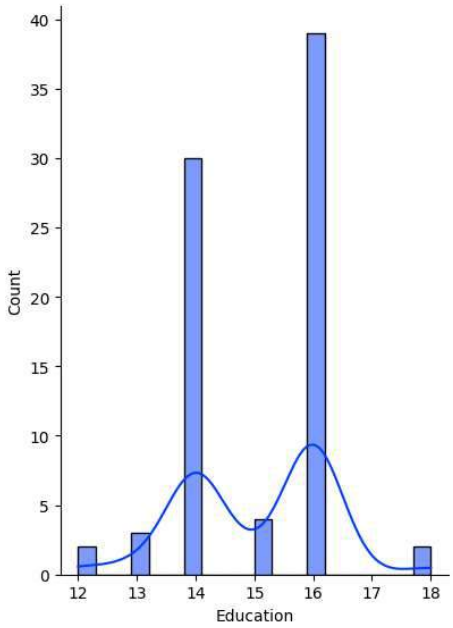
```
# saving the new file of raw cleaned data:
df.to_csv('aerofit_final.csv',sep=',',index=False)


p = df.Product.unique()
num_cols = ['Age','Education','Usage','Fitness','Income','Miles']

for i in range(len(num_cols)):
    fig,ax = plt.subplots(nrows=1, ncols=3, figsize=(15,6.5))
    plt.suptitle(f"Customers Preferrence of Products based on {num_cols[i]}",fontsize=20,fontfamily='serif',fontweight='bold'
                 ,color='w')
    for j in range(len(p)):
        prd = df[df.Product==p[j]]
        sns.histplot(data=prd, x=num_cols[i], bins=20, kde=True, ax=ax[j])
        sns.despine()
        ax[j].set_title(f"{p[j]}'s Customers with {num_cols[i]}",color='w')
```

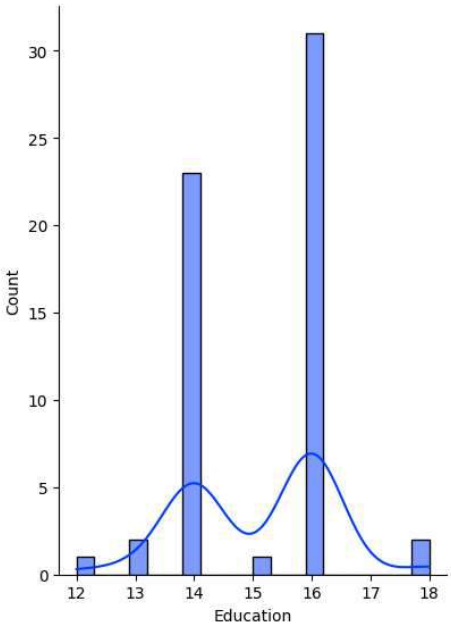
Web search Copy

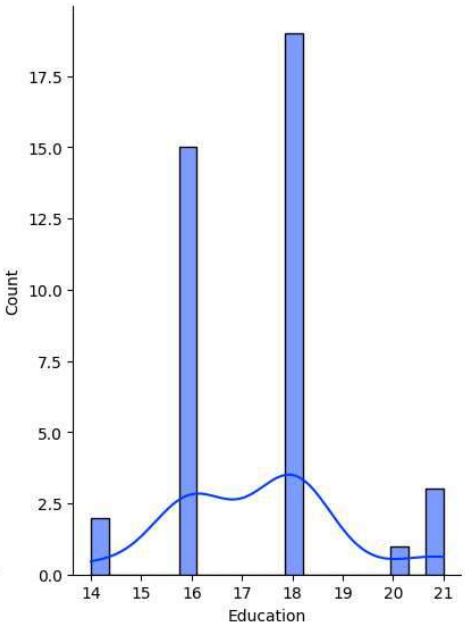


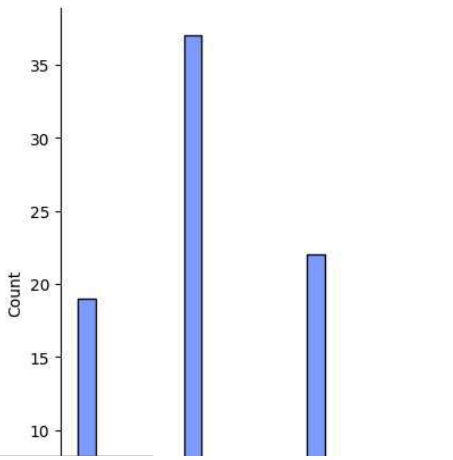


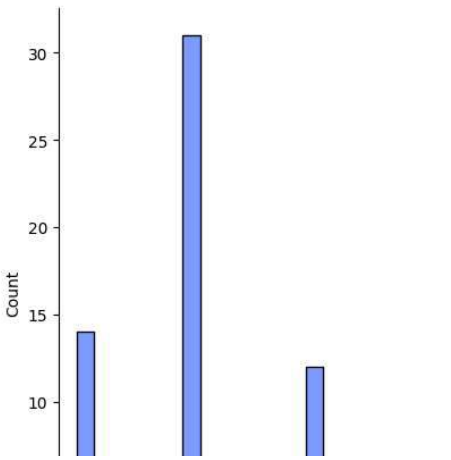


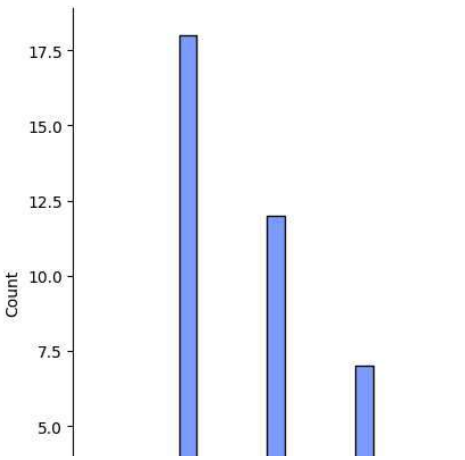


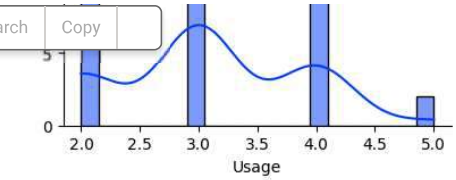


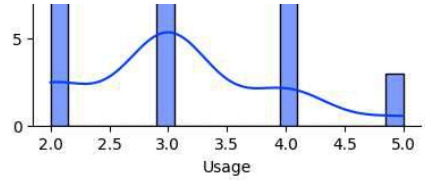


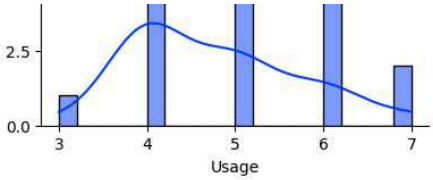


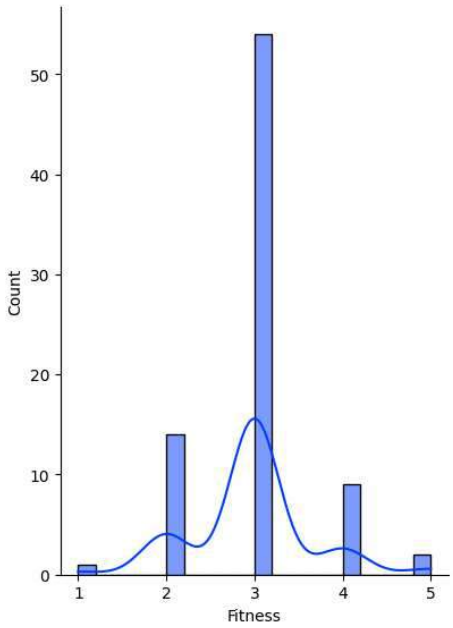


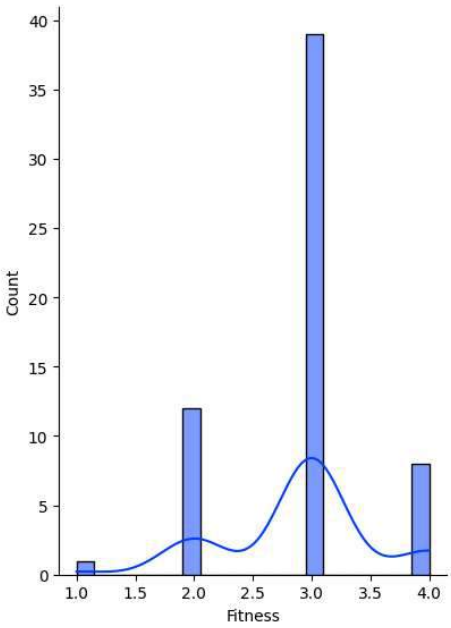


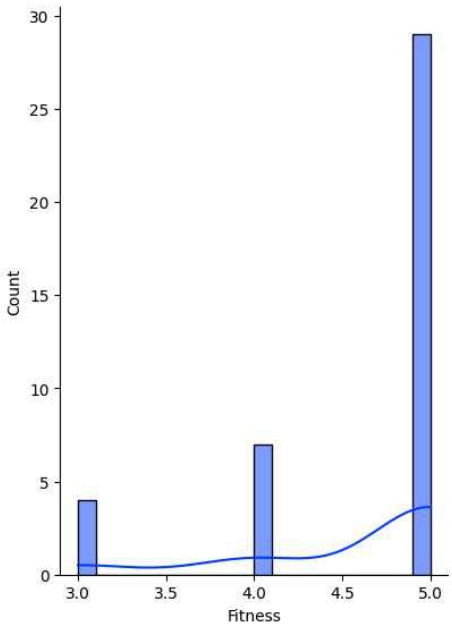


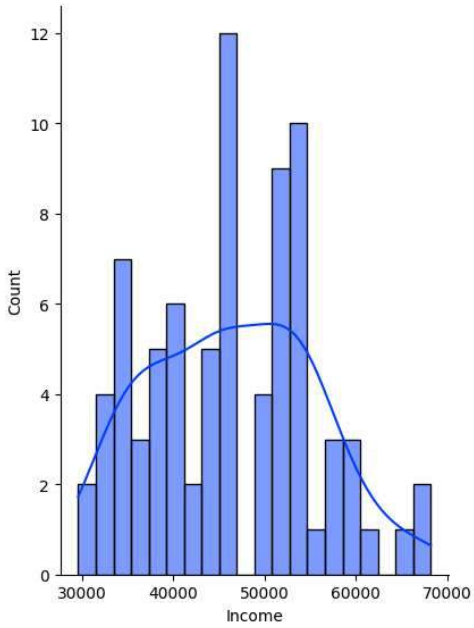


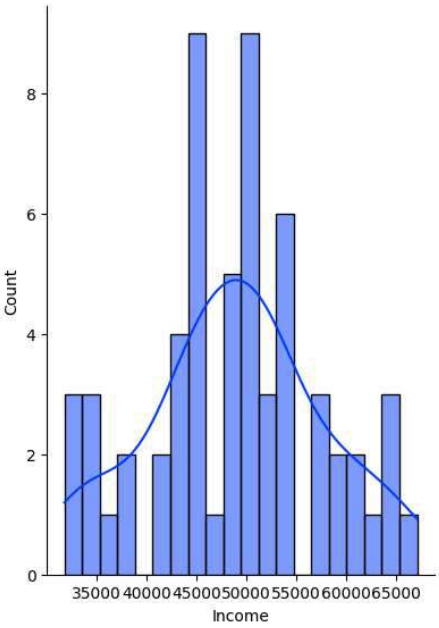


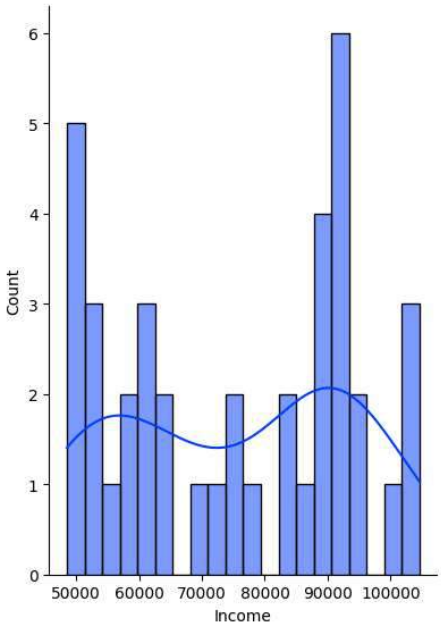


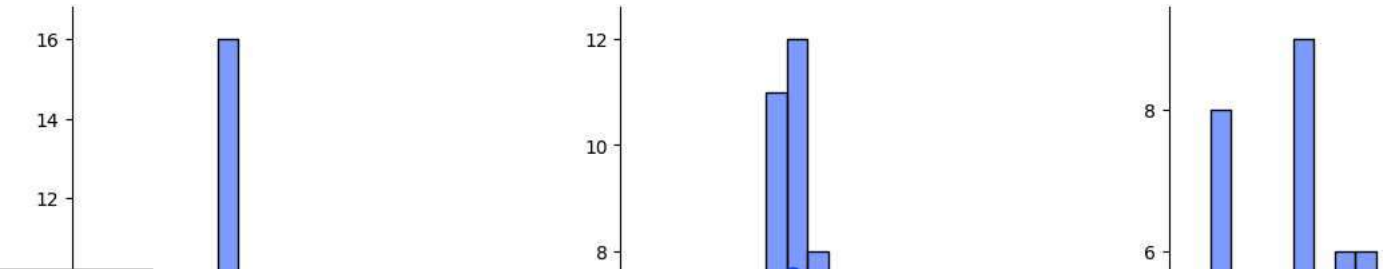


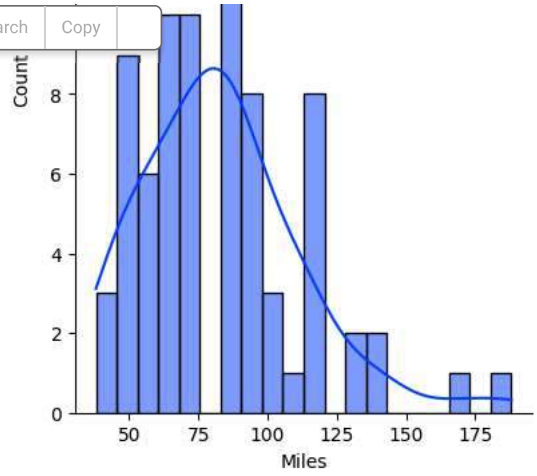


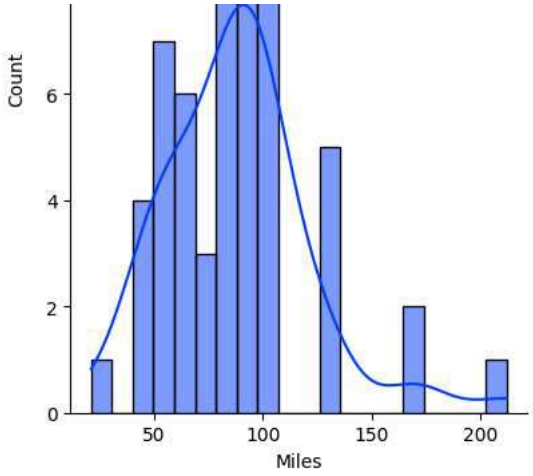


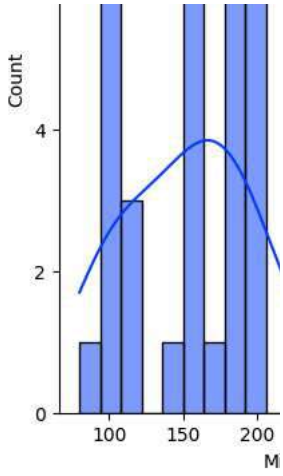












Web search Copy

- mers bought KP281
- 33.33% of customers bought KP481
- 22.22% of customers bought KP781

```
pms = df[(df['Gender']=='Male') & (df['MaritalStatus']=='Single')]
v1=round(len(pms[pms['Product']=='KP281'])/(len(pms))*100,2)
v2=round(len(pms[pms['Product']=='KP481'])/(len(pms))*100,2)
v3=round(len(pms[pms['Product']=='KP781'])/(len(pms))*100,2)
print('Probability of Male and Single for buying')
print(f'- KP281 is {v1}%\n- KP481 is {v2}%\n- KP781 is {v3}%')
```

```
Probability of Male and Single for buying
- KP281 is 44.19%
- KP481 is 23.26%
- KP781 is 32.56%
```

```
pmp = df[(df['Gender']=='Male') & (df['MaritalStatus']=='Partnered')]
v1=round(len(pmp[pmp['Product']=='KP281'])/(len(pmp))*100,2)
v2=round(len(pmp[pmp['Product']=='KP481'])/(len(pmp))*100,2)
v3=round(len(pmp[pmp['Product']=='KP781'])/(len(pmp))*100,2)
print('Probability of Male and Partnered for buying')
print(f'- KP281 is {v1}%\n- KP481 is {v2}%\n- KP781 is {v3}%')
```

```
Probability of Male and Partnered for buying
- KP281 is 34.43%
- KP481 is 34.43%
- KP781 is 31.15%
```

```
pfp = df[(df['Gender']=='Female') & (df['MaritalStatus']=='Partnered')]
v1=round(len(pfp[pfp['Product']=='KP281'])/(len(pfp))*100,2)
v2=round(len(pfp[pfp['Product']=='KP481'])/(len(pfp))*100,2)
v3=round(len(pfp[pfp['Product']=='KP781'])/(len(pfp))*100,2)
print('Probability of Female and Partnered for buying')
print(f'- KP281 is {v1}%\n- KP481 is {v2}%\n- KP781 is {v3}%')
```

```
Probability of Female and Partnered for buying
- KP281 is 58.7%
- KP481 is 32.61%
- KP781 is 8.7%
```

- Probability of Buying KP281 increased from 44.44% to 58.7%, if the customer is Female and Partnered.
- Probability of Buying KP481 increased from 33.33% to 46.67%, if the customer is Female and Single.
- Probability of Buying KP781 increased from 22.22% to 32.56%, if the customer is Male and Single.
- Probability of Buying KP481 & KP781 increased from 33.33% & 22.22% to 34.43%, if the customer is Male and Single.
- Probability of Buying KP781 decreased from 22.22% to 8.7%, if the customer is Female and Partnered.

```
mb = df.groupby(['MaritalStatus','Gender','Product']).agg(count=('Product','count'))
mb
```

Web search Copy

```
...7-e5771f06c922>:1: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future ve
mb = df.groupby(['MaritalStatus','Gender','Product']).agg(count=('Product','count'))
```

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

Next steps: Generate code with mb | View recommended plots | New interactive sheet

```
maf = df[['Product','Gender','MaritalStatus']].melt()
maf
```

	variable	value
0	Product	KP281
1	Product	KP281
2	Product	KP281
3	Product	KP281
4	Product	KP281
...	...	...
535	MaritalStatus	Single
536	MaritalStatus	Single
537	MaritalStatus	Single
538	MaritalStatus	Partnered
539	MaritalStatus	Partnered

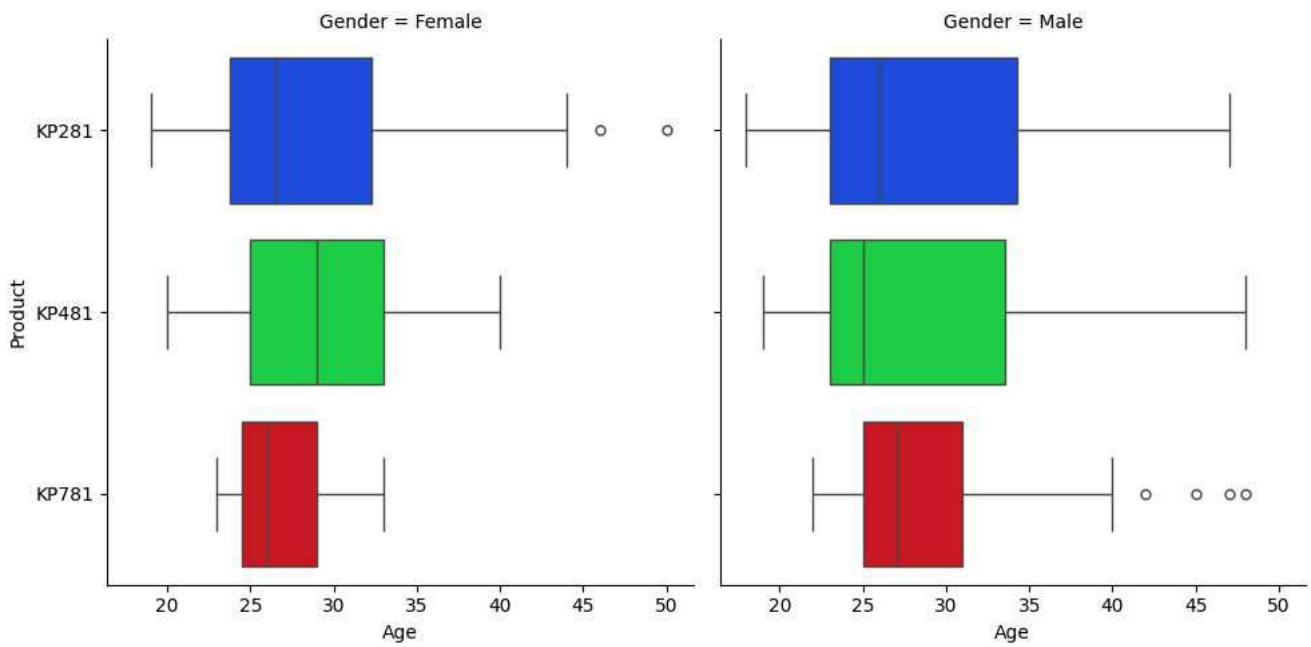
540 rows × 2 columns

Next steps: Generate code with maf | View recommended plots | New interactive sheet

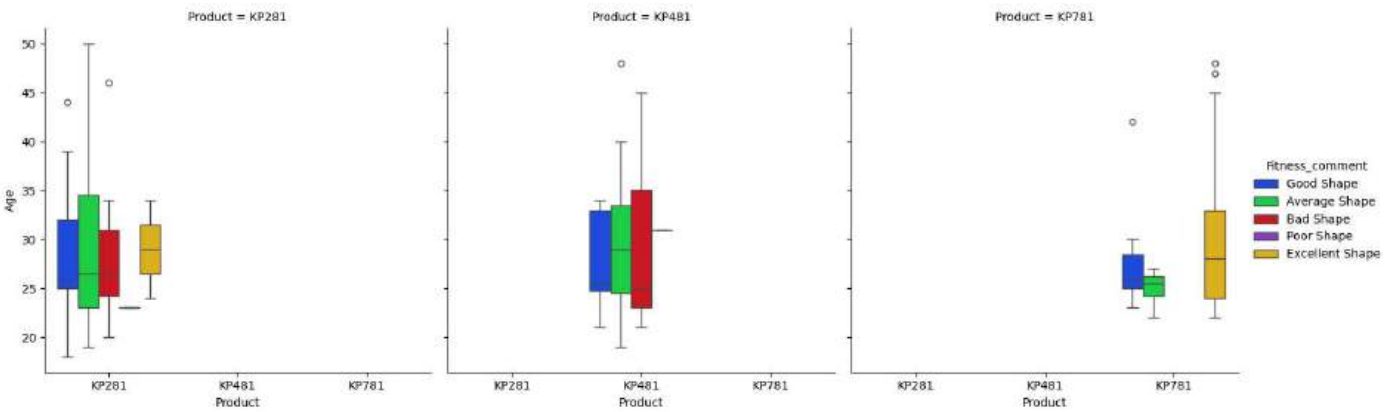
```
maff = maf.groupby(['variable','value']).agg(cust_cnt=('value','count'))
maff
```

		cust_cnt
variable	value	
Gender	Female	76
	Male	104
MaritalStatus	Partnered	107
	Single	73
Product	KP281	80
	KP481	60
	KP781	40

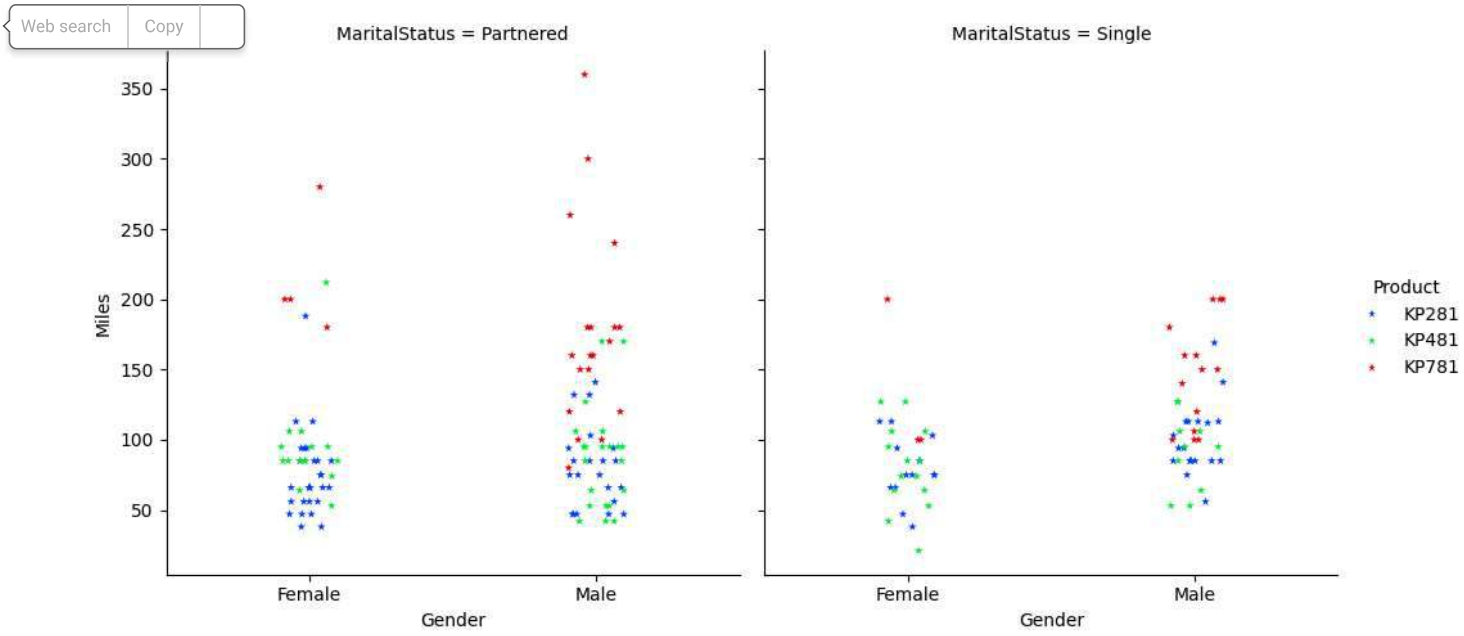
```
# Product used among age group seggregated by Gender
sns.catplot(x='Age',y='Product',hue='Product',col='Gender',data=df,kind='box')
plt.show()
```




```
# Product used among age group seggregated by Gender
sns.catplot(y='Age',x='Product',hue='Fitness_comment',col='Product',data=df,kind='box')
plt.show()
```




```
# Miles covered in each product by gender and their marital status

# sns.stripplot(x='Gender',y='Miles',hue='Product',data=af,palette=cp,marker='*')

sns.catplot(x='Gender',y='Miles',hue='Product',col='MaritalStatus',data=df,kind='strip',marker='*')
plt.show()
```



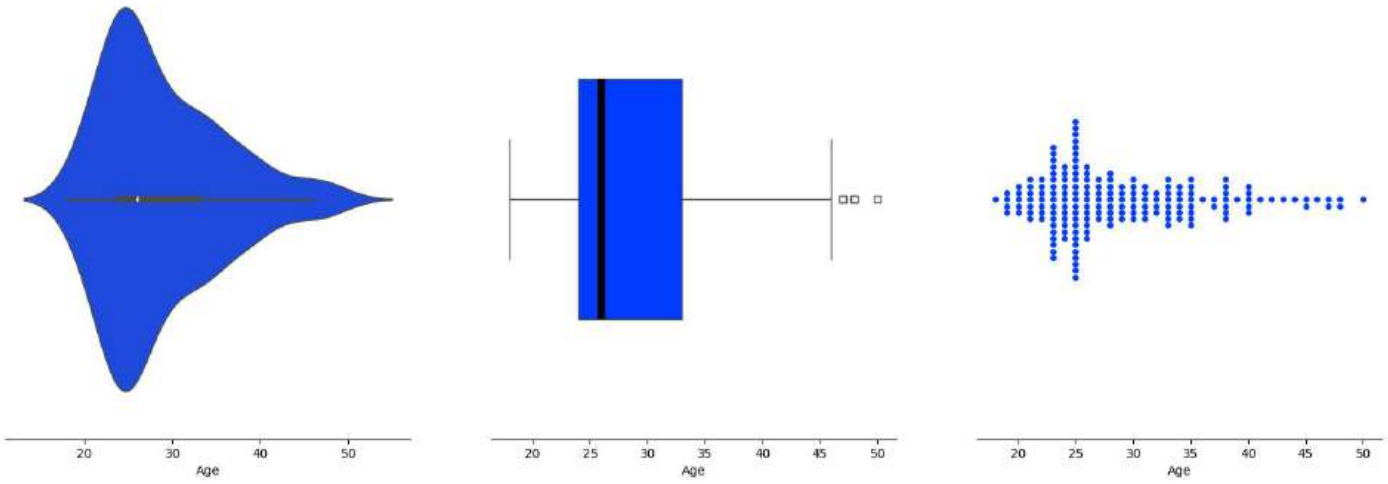

```
num_cols = ['Age','Education','Usage','Fitness','Income','Miles']

for i in range(len(num_cols)):
    data = df[num_cols[i]].tolist()
    mini = np.min(data)
    Q1 = np.percentile(data, 25)
    Q2 = np.median(data)
    Q3 = np.percentile(data, 75)
    maxi = np.max(data)
    IQR = Q3 - Q1

    lo = Q1 - (1.5 * IQR)
    ho = Q3 + (1.5 * IQR)

    lower_outliers=[]
    upper_outliers=[]
    for k in data:
        if k < lo:
            lower_outliers.append(k)

        elif k > ho:
            upper_outliers.append(k)

    uo_pct = round((len(upper_outliers)*100/df.shape[0]),2)
    lo_pct = round((len(lower_outliers)*100/df.shape[0]),2)

    print()
    print(f"Outlier detection of {num_cols[i]}")
    print('.'*30)
    print("Minimum:", mini)
    print("Maximum:", maxi)
    print(f'Initial Range (with outlier) : {(maxi-mini)}')
    print("Q1:", Q1)
    print("Q2:", Q2)
    print("Q3:", Q3)
    print("IQR:", IQR)
    print(f'Final Range (without outlier) : {(ho-lo)}')
    print("Lower outliers are:", lower_outliers)
    print("Upper outliers are:", upper_outliers)
    print(f'Lower Outlier Percentage is {lo_pct}%')
    print(f'Upper Outlier Percentage is {uo_pct}%')
    print(f'Overall Outlier Percentage is {(lo_pct+uo_pct)}%')

    if len(set(lower_outliers)):
        print(f'Outlier points towards left of boxplot : {len(set(lower_outliers))} and they are {(set(lower_outliers))}')
    else:
        print(f'Outlier points towards left of boxplot : {len(set(lower_outliers))}')
    if len(set(upper_outliers)):
        print(f'Outlier points towards right of boxplot : {len(set(upper_outliers))} and they are {(set(upper_outliers))}')
```

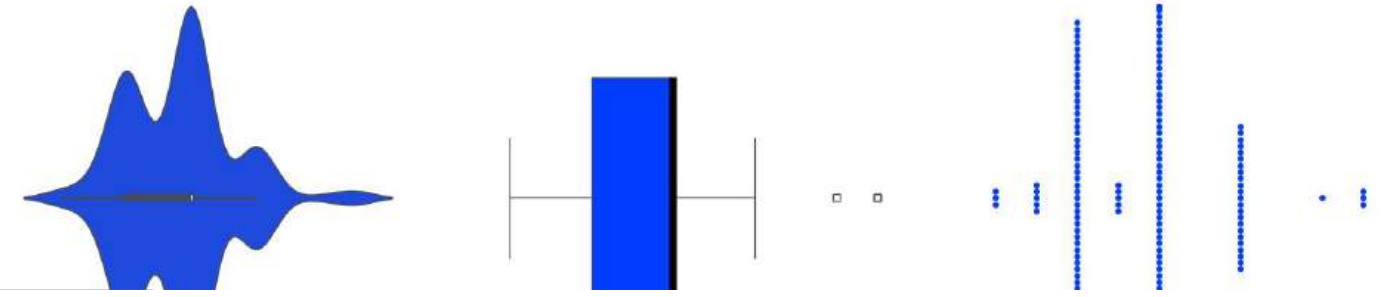
```
    print(f'Number of outliers points towards right of boxplot : {len(set(upper_outliers))}')
    print()

    plt.figure(figsize=(20,7))
    plt.style.use('default')
    plt.style.use('seaborn-v0_8-bright')
    plt.suptitle(f'Customers classification Based on {num_cols[i]}',fontfamily='serif',fontweight='bold',fontsize=20
              ,color='w')


    plt.subplot(1,3,1)
    sns.violinplot(df,x=num_cols[i])
    plt.title(f'Violinplot of {num_cols[i]}',fontfamily='serif',fontweight='bold',fontsize=12,
              loc='center',color='w')
    plt.yticks([])

    plt.subplot(1,3,2)
    bxp = sns.boxplot(df,x=num_cols[i],width=0.5,saturation=97,flierprops={"marker":"s"},
                      medianprops={"color": "k", "linewidth": 6})#,boxprops={"facecolor": (.3, .5, .7, .5)})
    plt.title(f'Box & Whisker plot of {num_cols[i]}',fontfamily='serif',fontweight='bold',fontsize=12,
              loc='center',color='w')
    sns.despine(left=True)
    plt.yticks([])

    plt.subplot(1,3,3)
    sns.swarmplot(df,x=num_cols[i])
    plt.title(f'Swarmplot of {num_cols[i]}',fontfamily='serif',fontweight='bold',fontsize=12,
              loc='center',color='w')
    sns.despine(left=True)
    plt.yticks([])
    print('-'*127)
```

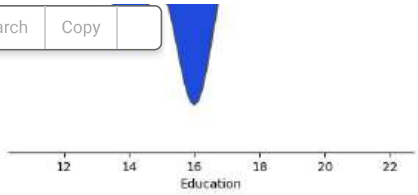
Web search Copy

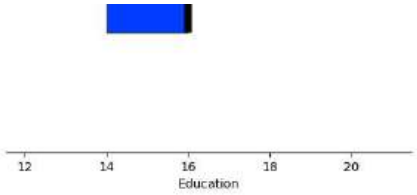
```
Outlier detection of Age
.............................
Minimum: 18
Maximum: 50
Initial Range (with outlier) : 32
Q1: 24.0
Q2: 26.0
Q3: 33.0
IQR: 9.0
Final Range (without outlier) : 36.0
Lower outliers are: []
Upper outliers are: [47, 50, 48, 47, 48]
Lower Outlier Percentage is 0.0%
Upper Outlier Percentage is 2.78%
Overall Outlier Percentage is 2.78%
Outlier points towards left of boxplot : 0
Outlier points towards right of boxplot : 3 and they are {48, 50, 47}

-------------------------------------------------------------------------------------------------------------------------------

Outlier detection of Education
.............................
Minimum: 12
Maximum: 21
Initial Range (with outlier) : 9
Q1: 14.0
Q2: 16.0
Q3: 16.0
IQR: 2.0
Final Range (without outlier) : 8.0
Lower outliers are: []
Upper outliers are: [20, 21, 21, 21]
Lower Outlier Percentage is 0.0%
Upper Outlier Percentage is 2.22%
Overall Outlier Percentage is 2.22%
Outlier points towards left of boxplot : 0
Outlier points towards right of boxplot : 2 and they are {20, 21}

-------------------------------------------------------------------------------------------------------------------------------

Outlier detection of Usage
.............................
Minimum: 2
Maximum: 7
Initial Range (with outlier) : 5
Q1: 3.0
Q2: 3.0
Q3: 4.0
IQR: 1.0
Final Range (without outlier) : 4.0
Lower outliers are: []
Upper outliers are: [6, 6, 6, 7, 6, 7, 6, 6, 6]
Lower Outlier Percentage is 0.0%
Upper Outlier Percentage is 5.0%
Overall Outlier Percentage is 5.0%
Outlier points towards left of boxplot : 0
Outlier points towards right of boxplot : 2 and they are {6, 7}

-------------------------------------------------------------------------------------------------------------------------------

Outlier detection of Fitness
.............................
Minimum: 1
Maximum: 5
Initial Range (with outlier) : 4
Q1: 3.0
Q2: 3.0
Q3: 4.0
IQR: 1.0
Final Range (without outlier) : 4.0
Lower outliers are: [1, 1]
Upper outliers are: []
Lower Outlier Percentage is 1.11%
Upper Outlier Percentage is 0.0%
Overall Outlier Percentage is 1.11%
Outlier points towards left of boxplot : 1 and they are {1}
Outlier points towards right of boxplot : 0

-------------------------------------------------------------------------------------------------------------------------------

Outlier detection of Income
.............................
Minimum: 29562
```

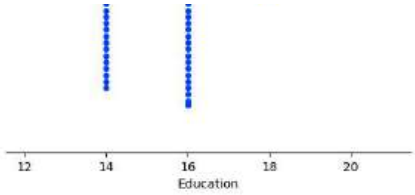
```
ith outlier) : 75019
Q1: 44058.75
Q2: 50596.5
Q3: 58668.0
IQR: 14609.25
Final Range (without outlier) : 58437.0
Lower outliers are: []
Upper outliers are: [83416, 88396, 90886, 92131, 88396, 85906, 90886, 103336, 99601, 89641, 95866, 92131, 92131, 104581, 83416, 89641, 9
Lower Outlier Percentage is 0.0%
Upper Outlier Percentage is 10.56%
Overall Outlier Percentage is 10.56%
Outlier points towards left of boxplot : 0
Outlier points towards right of boxplot : 11 and they are {92131, 104581, 90886, 103336, 89641, 88396, 99601, 85906, 95508, 83416, 95866

-------------------------------------------------------------------------------------------------------------------------------------

Outlier detection of Miles
..........................
Minimum: 21
Maximum: 360
Initial Range (with outlier) : 339
Q1: 66.0
Q2: 94.0
Q3: 114.75
IQR: 48.75
Final Range (without outlier) : 195.0
Lower outliers are: []
Upper outliers are: [188, 212, 200, 200, 200, 240, 300, 280, 260, 200, 360, 200, 200]
Lower Outlier Percentage is 0.0%
Upper Outlier Percentage is 7.22%
Overall Outlier Percentage is 7.22%
Outlier points towards left of boxplot : 0
Outlier points towards right of boxplot : 8 and they are {260, 200, 360, 300, 240, 212, 280, 188}

-------------------------------------------------------------------------------------------------------------------------------------
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:3399: UserWarning: 14.4% of the points cannot be placed; you may want to
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:3399: UserWarning: 5.6% of the points cannot be placed; you may want to d
  warnings.warn(msg, UserWarning)
/usr/local/lib/python3.10/dist-packages/seaborn/categorical.py:3399: UserWarning: 21.1% of the points cannot be placed; you may want to
  warnings.warn(msg, UserWarning)
```

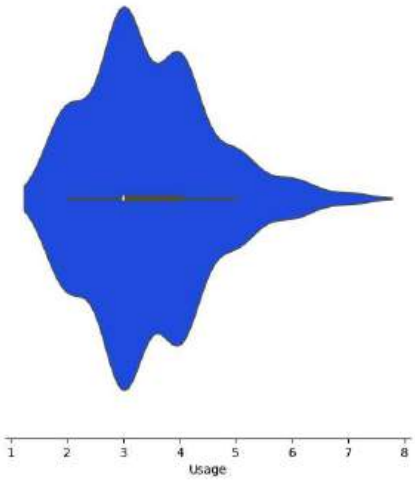


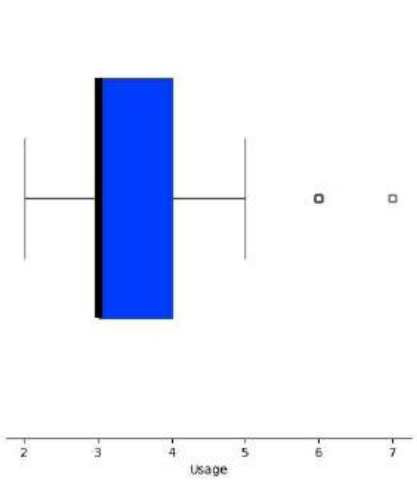


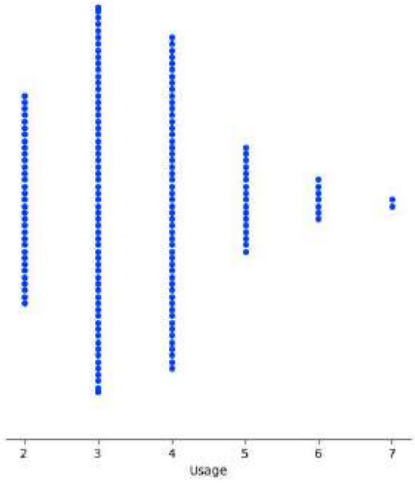


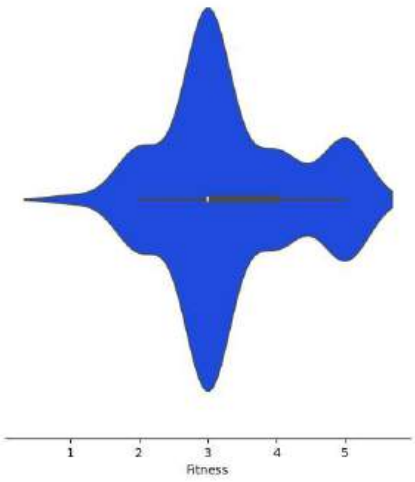


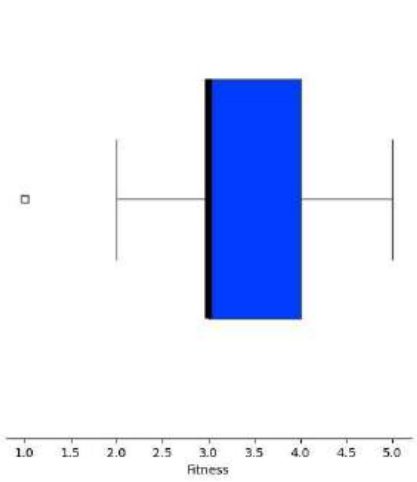


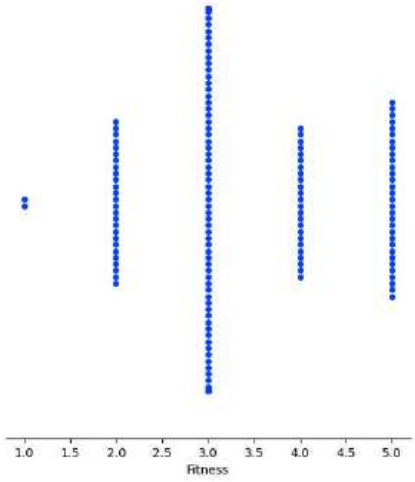


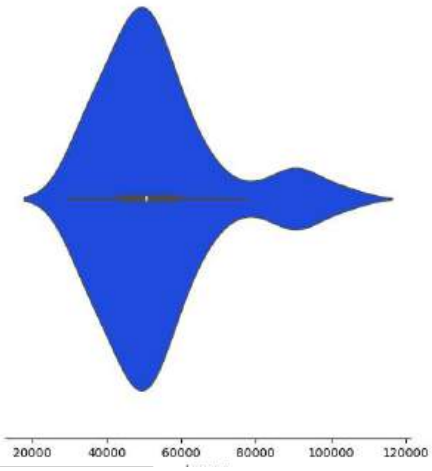


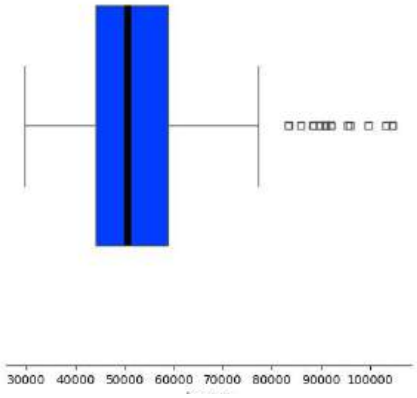


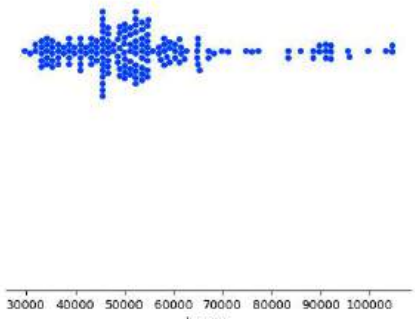


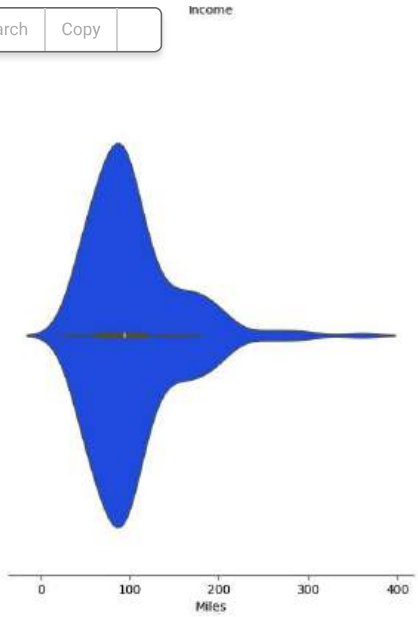


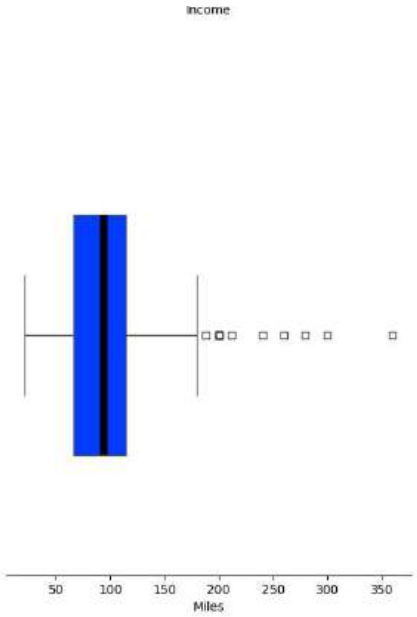


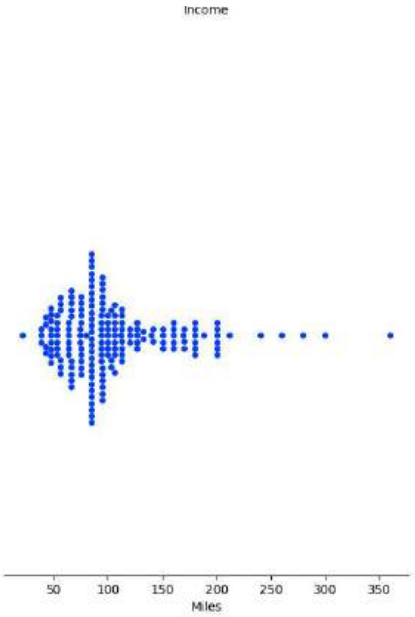






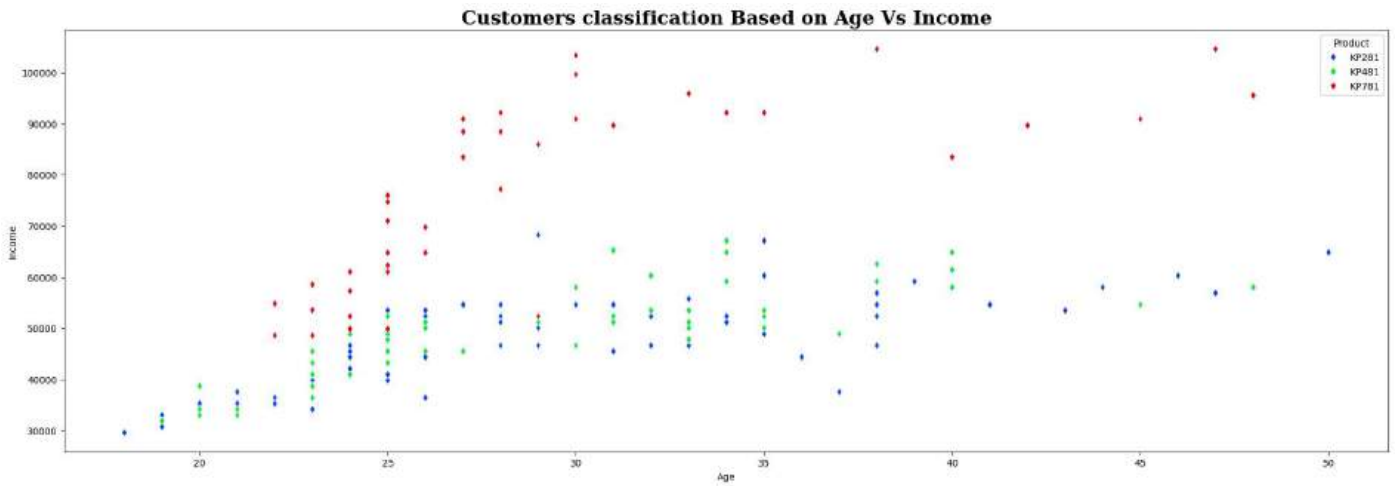




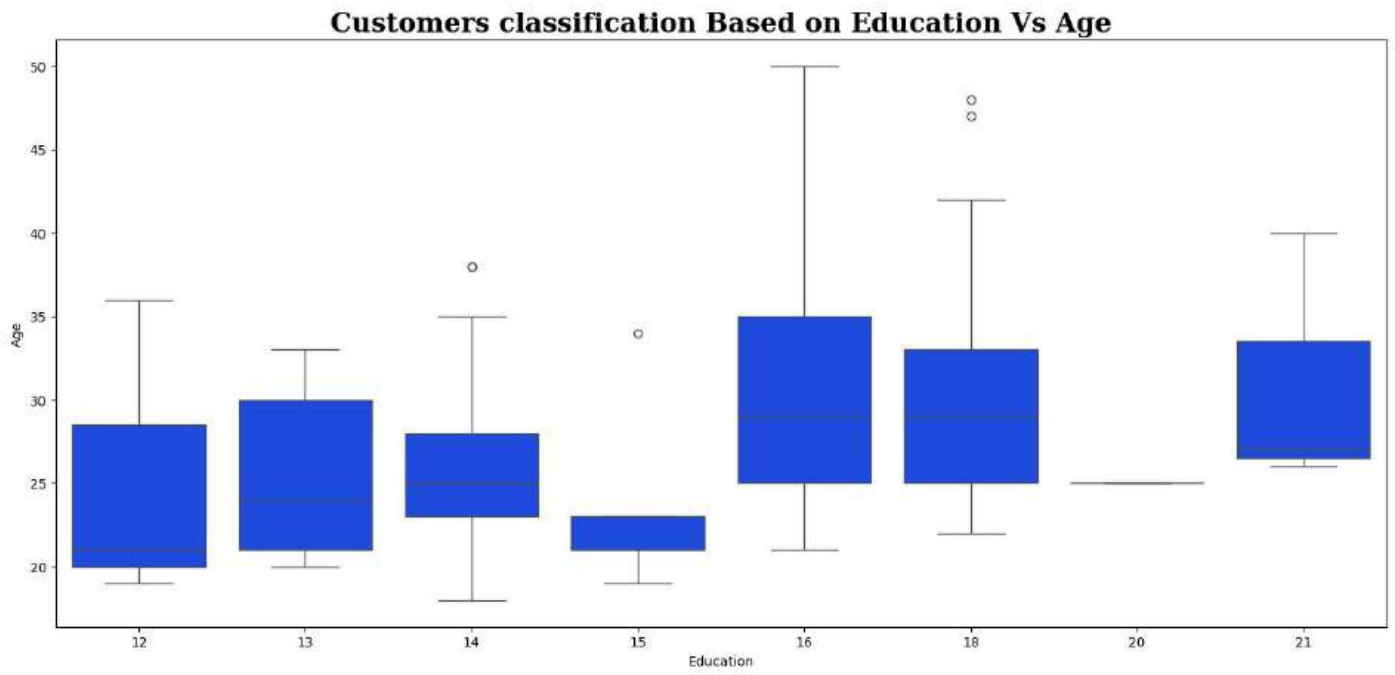


Web search Copy

```
plt.figure(figsize=(18,8))
sns.scatterplot(data=df, x='Age',y='Income', hue='Product',marker='d')
plt.title('Customers classification Based on Age Vs Income',fontfamily='serif',fontweight='bold',fontsize=20)
plt.show()
```

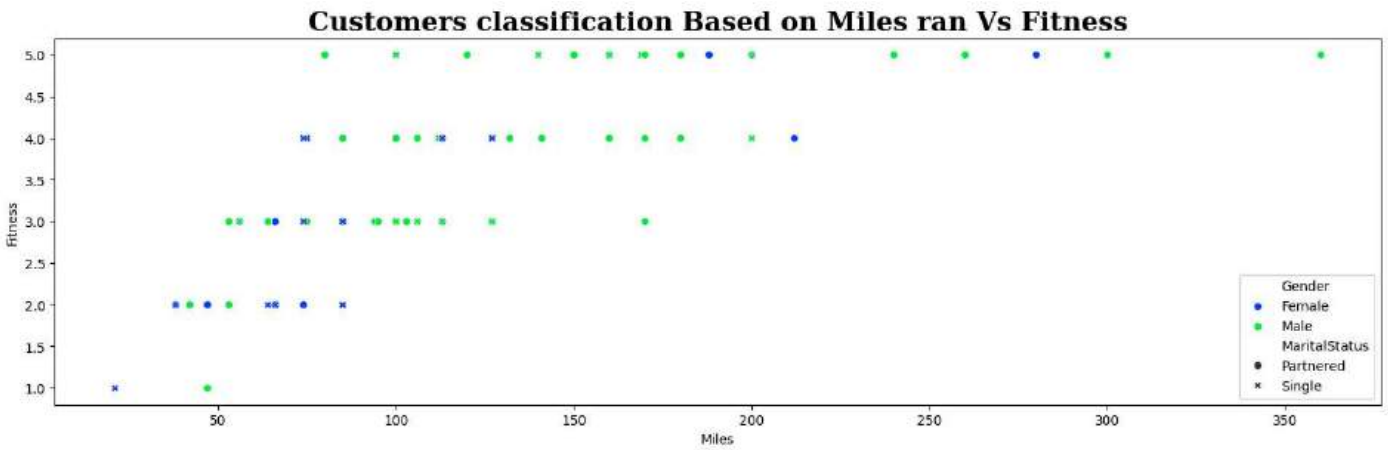



```
plt.figure(figsize=(18,8))
sns.boxplot(df,x='Education',y='Age')
plt.title('Customers classification Based on Education Vs Age',fontfamily='serif',fontweight='bold',fontsize=20)
plt.show()
```

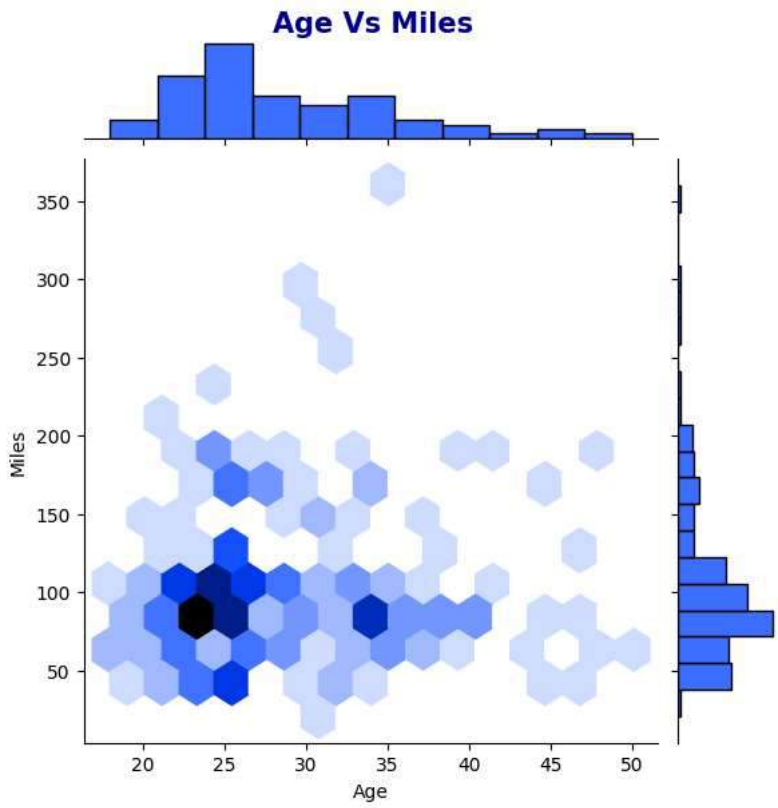



```
plt.figure(figsize=(18,5))
sns.scatterplot(x='Miles',y='Fitness',data=df,hue='Gender',style='MaritalStatus')
```

```
Web search  Copy
classification Based on Miles ran Vs Fitness',fontfamily='serif',fontweight='bold',fontsize=20)
plt.legend(loc='lower right')
plt.show()
```

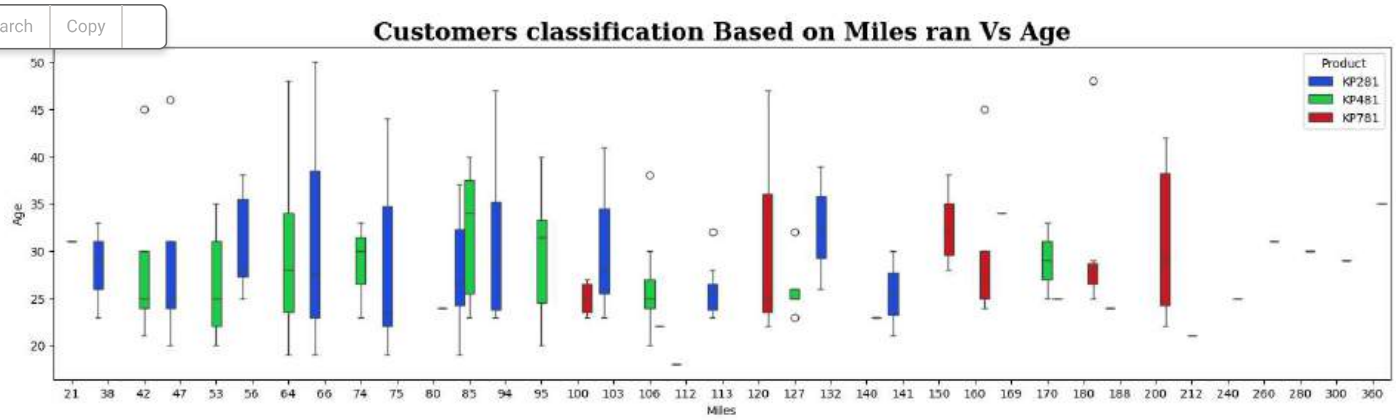



```
sns.jointplot(x="Age", y="Miles",data=df,kind="hex")
plt.text(28,458,"Age Vs Miles",color='darkblue',fontsize=15,fontweight='bold')
plt.show()
```

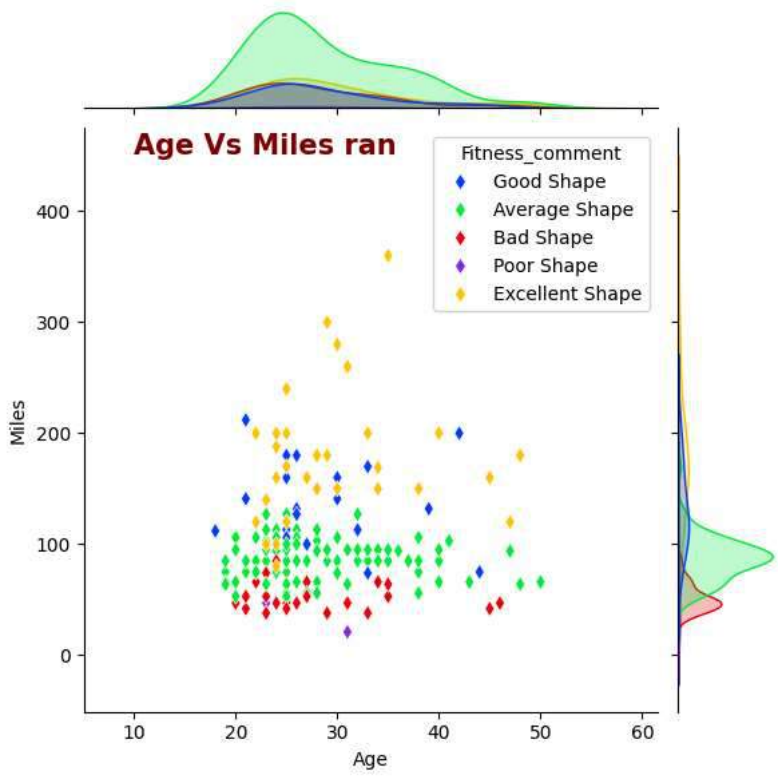



```
# Miles with each product
plt.figure(figsize=(20,5))
sns.boxplot(df,x='Miles',y='Age',hue='Product')
plt.title('Customers classification Based on Miles ran Vs Age',fontfamily='serif',fontweight='bold',fontsize=20)
plt.show()
print()
```

Web search Copy




```
sns.jointplot(x="Age", y="Miles",hue='Fitness_comment', data=df,marker='d')
plt.text(10,451,"Age Vs Miles ran",color='maroon',fontsize=15,fontweight='bold')
plt.show()
print()
```



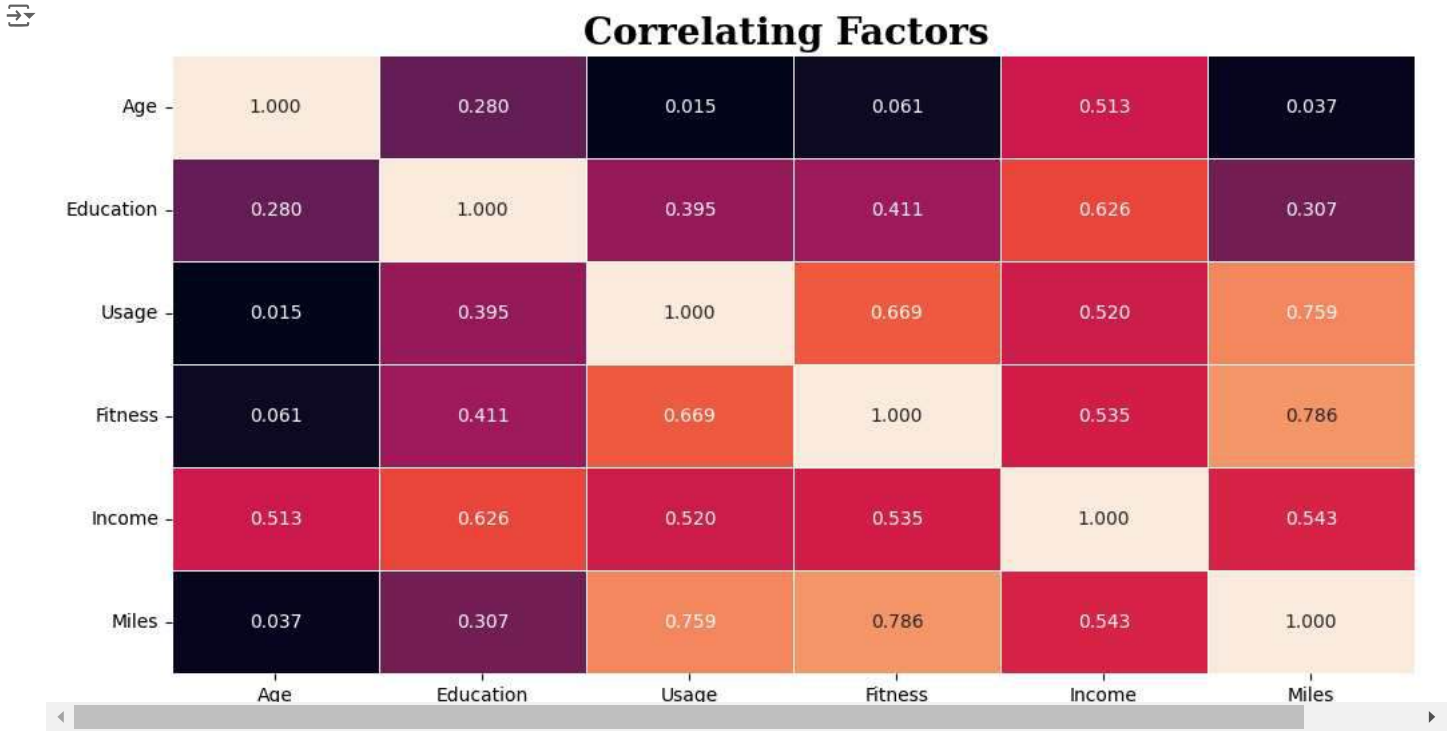

```
afcorr = df[['Age','Education','Usage','Fitness','Income','Miles']].corr()
afcorr
```

Web search | Copy

	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

Next steps: Generate code with afcorr | View recommended plots | New interactive sheet

```
#Correlation HeatMap
plt.figure(figsize=(15,6))
ax = sns.heatmap(afcorr,annot=True,fmt='.3f',linewidths=.5)
plt.title('Correlating Factors ',fontfamily='serif',fontweight='bold',fontsize=20)
plt.yticks(rotation=0)
plt.show()
```



From the pair plot we can see Age and Income are positively correlated and heatmap also suggests a strong correlation betwwen them

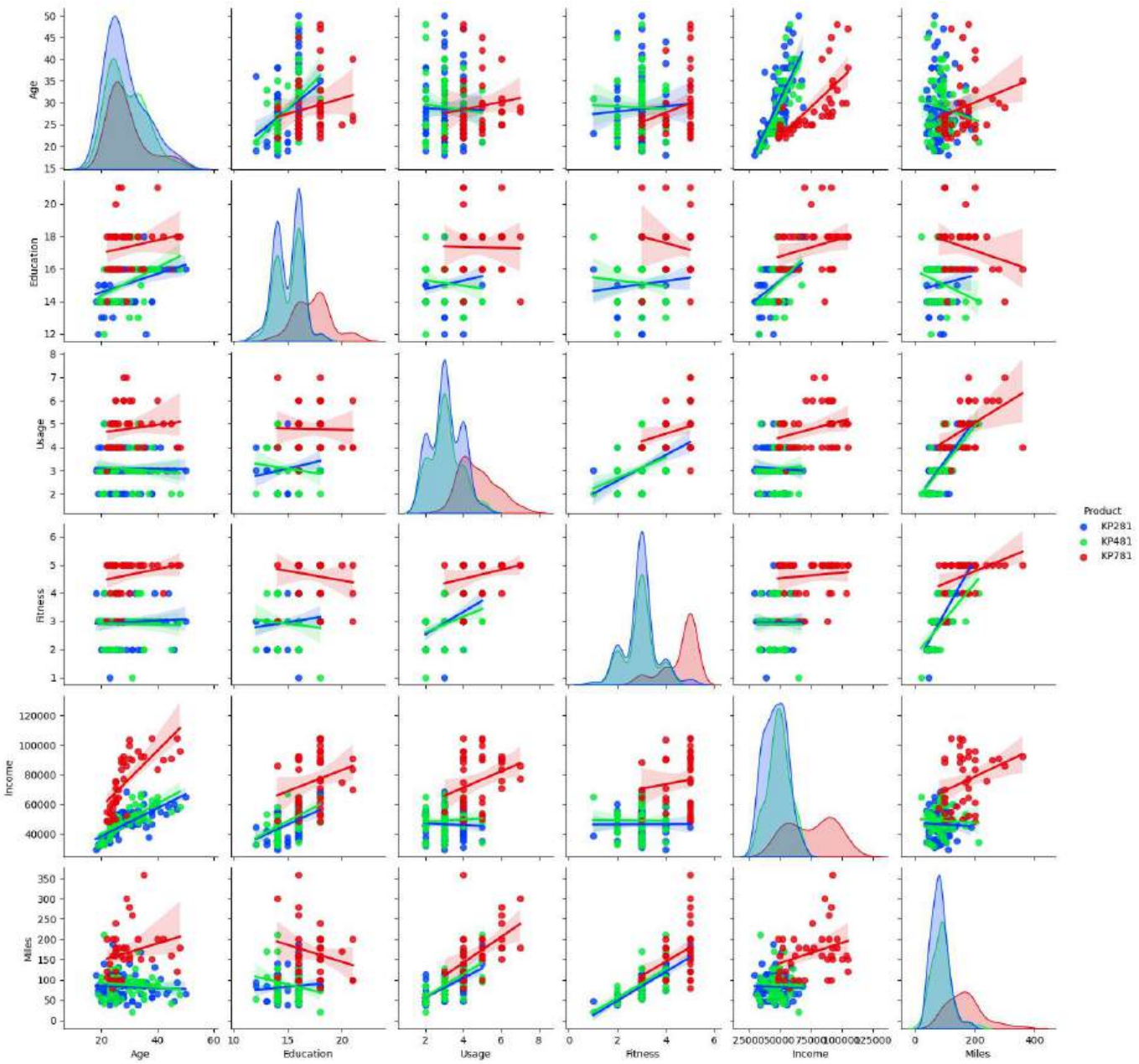
Eductaion and Income are highly correlated as its obvious. Eductation also has significatnt correlation between Fitness rating and Usage of the treadmill.

Usage is highly correlated with Fitness and Miles as more the usage more the fitness and mileage.

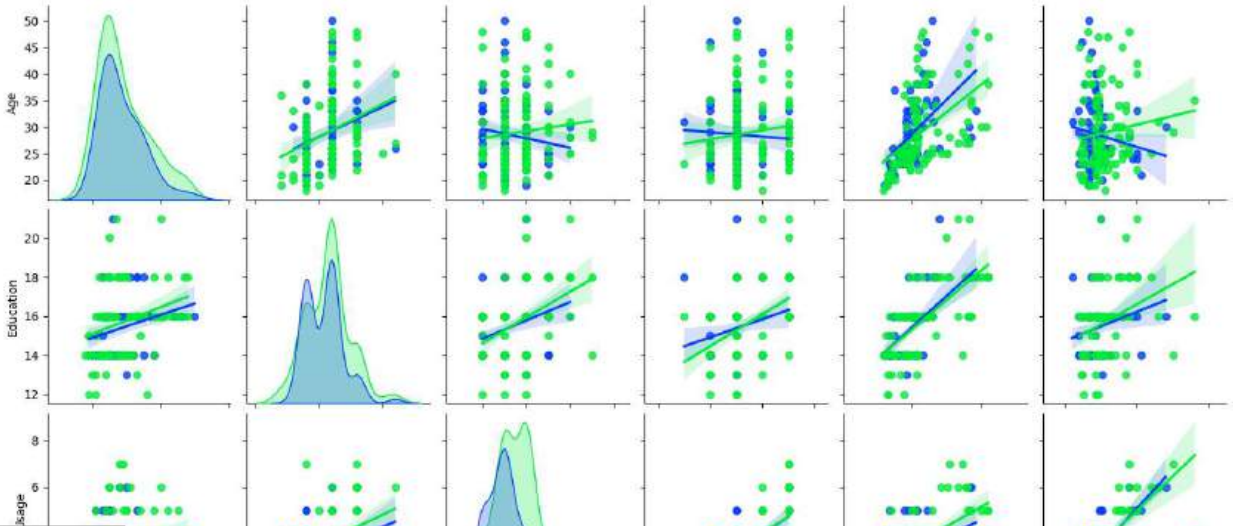
```
# Pairplot Analysis
cat_cols = ['Product','Gender','MaritalStatus']
for i in range(len(cat_cols)):
    plt.figure(figsize=(14,0.05))
    plt.axis('off')
    plt.title(f' Pairplot based on {cat_cols[i]} ',fontfamily='serif',fontweight='bold',fontsize=20,backgroundcolor='maroon',color='w')
    sns.pairplot(df,hue=cat_cols[i],kind='reg')
    plt.show()
    print()
```

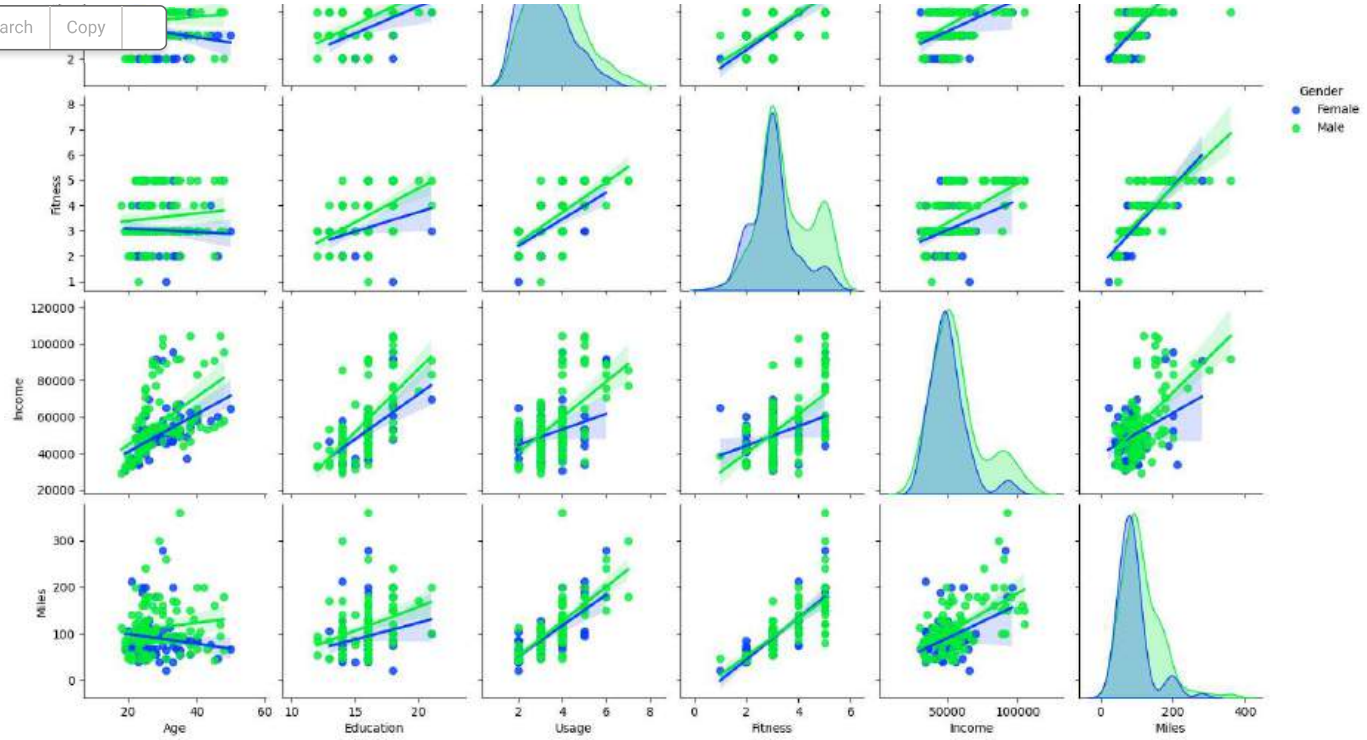
Web search Copy

Pairplot based on Product



Pairplot based on Gender





Pairplot based on MaritalStatus