AeroFit_Scaler

New notebook

In [46]:
```
# Welcome to your new notebook
# Type here in the cell editor to add code!
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset
df = pd.read_csv(f"{mssparkutils.nbResPath}/builtin/aerofit_treadmill.csv")
```

```
StatementMeta(, 878a4916-1406-44a1-82e3-1c7cbe560263, 48, Finished, Available, Finished)
```

Analysing basic metrics

In [47]:
```
# Display the first few rows of the dataframe
df.sample(10)
```

```
StatementMeta(, 878a4916-1406-44a1-82e3-1c7cbe560263, 49, Finished, Available, Finished)
```

Out[47]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
16	KP281	23	Female	14	Single	2	3	34110	103
81	KP481	20	Male	14	Single	2	3	32973	53
126	KP481	34	Male	16	Partnered	3	4	59124	85
43	KP281	27	Female	14	Partnered	2	3	45480	56
100	KP481	25	Female	14	Partnered	5	3	47754	106
152	KP781	25	Female	18	Partnered	5	5	61006	200
145	KP781	23	Male	16	Single	4	5	48556	100
75	KP281	43	Male	16	Partnered	3	3	53439	66
133	KP481	38	Female	16	Partnered	4	3	62535	85
49	KP281	28	Female	16	Partnered	3	3	51165	56

In [48]:

```
df['Product'].unique
```

```
StatementMeta(, 878a4916-1406-44a1-82e3-1c7cbe560263, 50, Finished, Available, Finished)
```

Out[48]:

```
<bound method Series.unique of 0      KP281
1      KP281
2      KP281
3      KP281
4      KP281
       ...
175    KP781
176    KP781
177    KP781
178    KP781
179    KP781
Name: Product, Length: 180, dtype: object>
```

In [49]:

```
# Check the structure and data types
df.info()
```

```
StatementMeta(, 878a4916-1406-44a1-82e3-1c7cbe560263, 51, Finished, Available, Finished)
```

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

Observations on the shape of data, data types of all the attributes, missing value detection, statistical summary

In [50]:
```
# Statistical summary of the dataset
df.describe()

# Check for missing values
df.isnull().sum()
```

```
StatementMeta(, 878a4916-1406-44a1-82e3-1c7cbe560263, 52, Finished, Available, Finished)
```

Out[50]:
```
Product          0
Age              0
Gender           0
Education        0
MaritalStatus    0
Usage            0
Fitness          0
Income           0
Miles            0
dtype: int64
```

In [51]:
```
#changing column types objects to category
categorical_cols = ['Product', 'Gender', 'MaritalStatus']
```

```
df[categorical_cols] = df[categorical_cols].astype('category')
df.dtypes
```

```
StatementMeta(, 878a4916-1406-44a1-82e3-1c7cbe560263, 53, Finished, Available, Finished)
```

Out[51]:

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

Non-Graphical Analysis: Value Counts and Unique Attributes

In [52]:

```
Age_range = df['Age'].min(), df['Age'].max()
print(f"Range of Age: {Age_range}")
Income_range = df['Income'].min(), df['Income'].max()
print(f"Range of Income: {Income_range}")
```

```
StatementMeta(, 878a4916-1406-44a1-82e3-1c7cbe560263, 54, Finished, Available, Finished)
Range of Age: (18, 50)
Range of Income: (29562, 104581)
```

In [53]:

```
# Value counts for categorical variables
product_counts = df['Product'].value_counts()
gender_counts = df['Gender'].value_counts()
marital_status_counts = df['MaritalStatus'].value_counts()

product_counts, gender_counts, marital_status_counts
```

```
StatementMeta(, 878a4916-1406-44a1-82e3-1c7cbe560263, 55, Finished, Available, Finished)
```

Out[53]:
```
(Product
 KP281    80
 KP481    60
 KP781    40
 Name: count, dtype: int64,
 Gender
 Male      104
 Female     76
 Name: count, dtype: int64,
 MaritalStatus
 Partnered    107
 Single        73
 Name: count, dtype: int64)
```

In [54]:
```
df.describe(include='all')
```

```
StatementMeta(, 878a4916-1406-44a1-82e3-1c7cbe560263, 56, Finished, Available, Finished)
```

Out[54]:

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles
count	180	180.000000	180	180.000000	180	180.000000	180.000000	180.000000	180.000000
unique	3	NaN	2	NaN	2	NaN	NaN	NaN	NaN
top	KP281	NaN	Male	NaN	Partnered	NaN	NaN	NaN	NaN
freq	80	NaN	104	NaN	107	NaN	NaN	NaN	NaN
mean	NaN	28.788889	NaN	15.572222	NaN	3.455556	3.311111	53719.577778	103.194444
std	NaN	6.943498	NaN	1.617055	NaN	1.084797	0.958869	16506.684226	51.863605
min	NaN	18.000000	NaN	12.000000	NaN	2.000000	1.000000	29562.000000	21.000000
25%	NaN	24.000000	NaN	14.000000	NaN	3.000000	3.000000	44058.750000	66.000000
50%	NaN	26.000000	NaN	16.000000	NaN	3.000000	3.000000	50596.500000	94.000000
75%	NaN	33.000000	NaN	16.000000	NaN	4.000000	4.000000	58668.000000	114.750000
max	NaN	50.000000	NaN	21.000000	NaN	7.000000	5.000000	104581.000000	360.000000

```
In [55]: # Group by Product to get summary statistics for each category
profile = df.groupby('Product').agg({
    'Age': ['mean', 'median'],
    'Income': ['mean', 'median'],
    'Usage': ['mean', 'median'],
    'Miles': ['mean', 'median'],
    'Fitness': ['mean', 'median']
})
print(profile)
```

```
StatementMeta(, 878a4916-1406-44a1-82e3-1c7cbe560263, 57, Finished, Available, Finished)
           Age            Income                 Usage                Miles         \
          mean median       mean    median       mean median         mean median
Product
KP281    28.55   26.0  46418.025  46617.0  3.087500    3.0   82.787500   85.0
KP481    28.90   26.0  48973.650  49459.5  3.066667    3.0   87.933333   85.0
KP781    29.10   27.0  75441.575  76568.5  4.775000    5.0  166.900000  160.0

        Fitness
           mean median
Product
KP281    2.9625    3.0
KP481    2.9000    3.0
KP781    4.6250    5.0
```

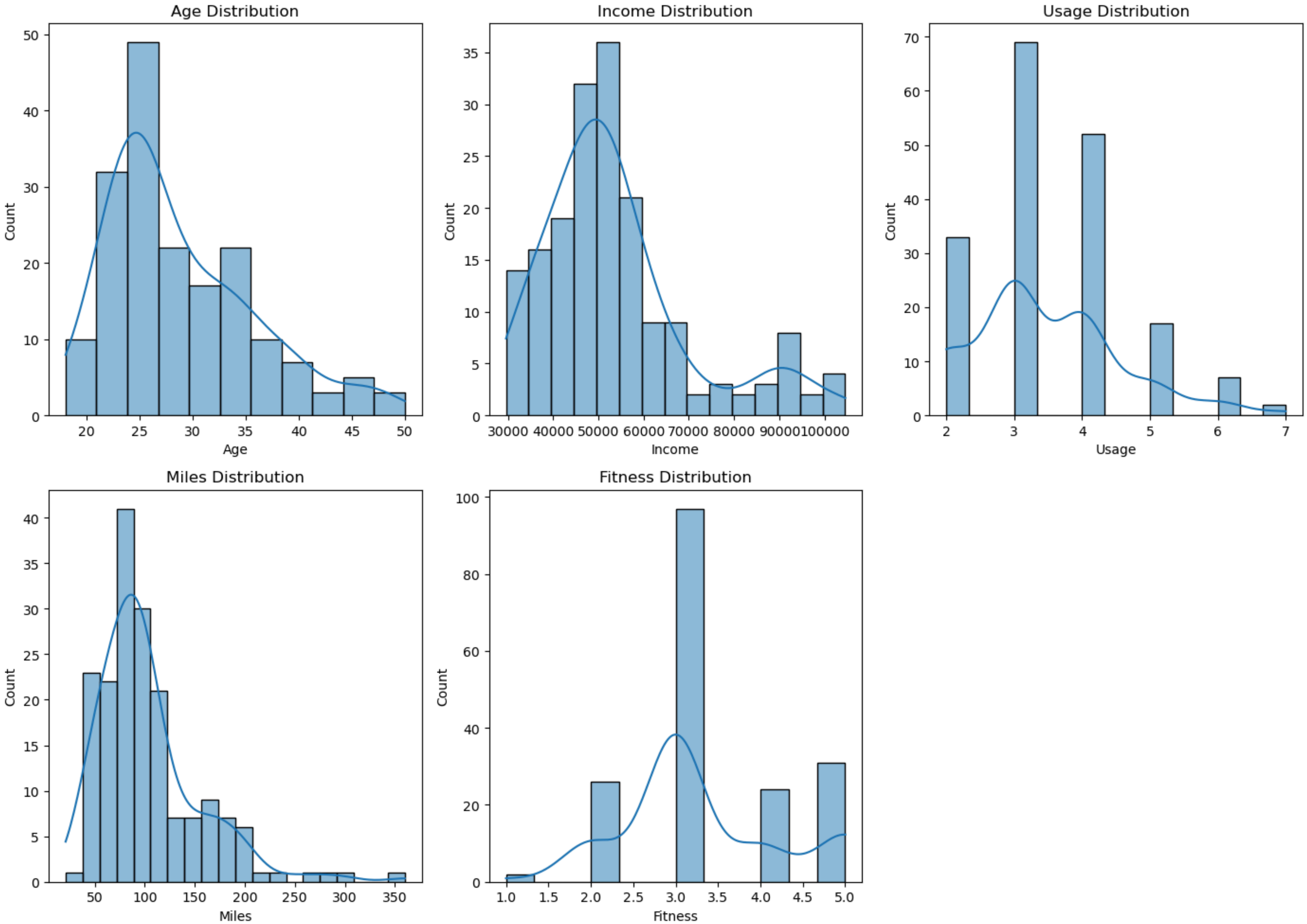
Visual Analysis - Univariate & Bivariate

Univariate Analysis for Continuous Variables

```
In [56]: # Univariate Analysis - Continuous Variables
plt.figure(figsize=(14, 10))

# Age Distribution
plt.subplot(2, 3, 1)
sns.histplot(df['Age'], kde=True)
plt.title('Age Distribution')

# Income Distribution
plt.subplot(2, 3, 2)
```

```
sns.histplot(df['Income'], kde=True)
plt.title('Income Distribution')

# Usage Distribution
plt.subplot(2, 3, 3)
sns.histplot(df['Usage'], kde=True)
plt.title('Usage Distribution')

# Miles Distribution
plt.subplot(2, 3, 4)
sns.histplot(df['Miles'], kde=True)
plt.title('Miles Distribution')

# Fitness Distribution
plt.subplot(2, 3, 5)
sns.histplot(df['Fitness'], kde=True)
plt.title('Fitness Distribution')

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

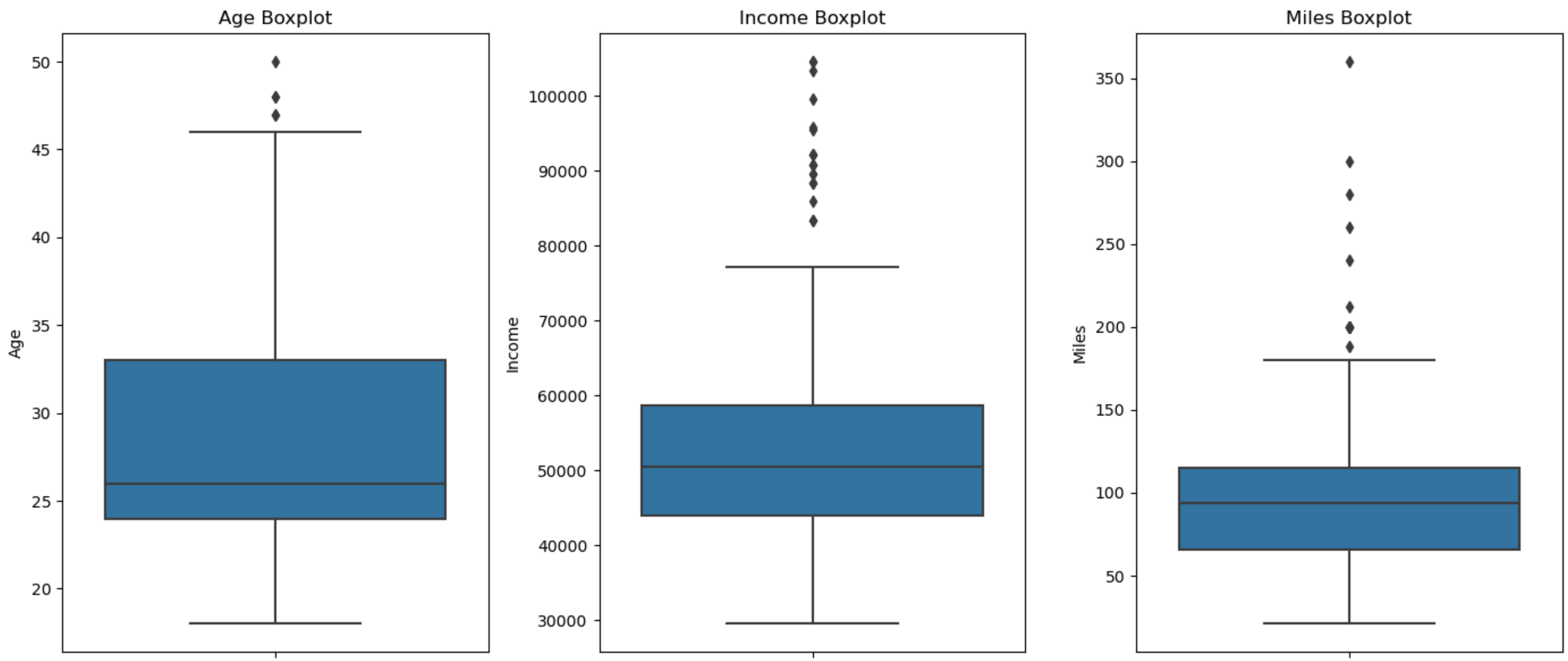
```
StatementMeta(, 878a4916-1406-44a1-82e3-1c7cbe560263, 58, Finished, Available, Finished)
```


Age Distribution
Count
Age
20
25
30
35
40
45
50
0
10
20
30
40
50
Income Distribution
Count
Income
30000
40000
50000
60000
70000
80000
90000
100000
0
5
10
15
20
25
30
35
Usage Distribution
Count
Usage
2
3
4
5
6
7
0
10
20
30
40
50
60
70
Miles Distribution
Count
Miles
50
100
150
200
250
300
350
0
5
10
15
20
25
30
35
40
Fitness Distribution
Count
Fitness
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
0
20
40
60
80
100

Boxplots for Outliers

In [57]:
```
# Boxplots for outliers
plt.figure(figsize=(14, 6))

# Age Boxplot
plt.subplot(1, 3, 1)
sns.boxplot(y=df['Age'])
plt.title('Age Boxplot')

# Income Boxplot
plt.subplot(1, 3, 2)
sns.boxplot(y=df['Income'])
plt.title('Income Boxplot')

# Miles Boxplot
plt.subplot(1, 3, 3)
sns.boxplot(y=df['Miles'])
plt.title('Miles Boxplot')

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

```
StatementMeta(, 878a4916-1406-44a1-82e3-1c7cbe560263, 59, Finished, Available, Finished)
```



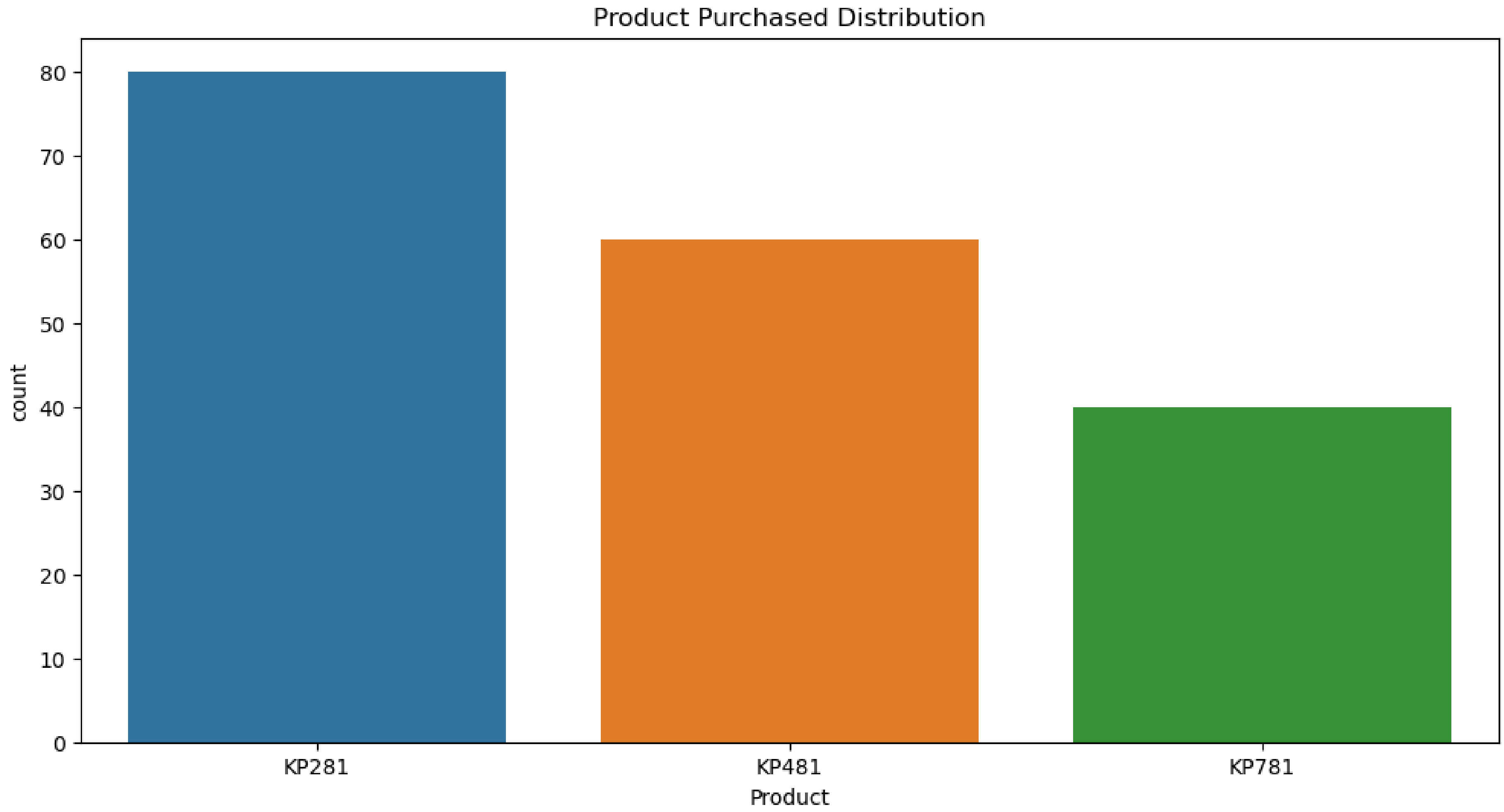
Count plots for Categorical Variables

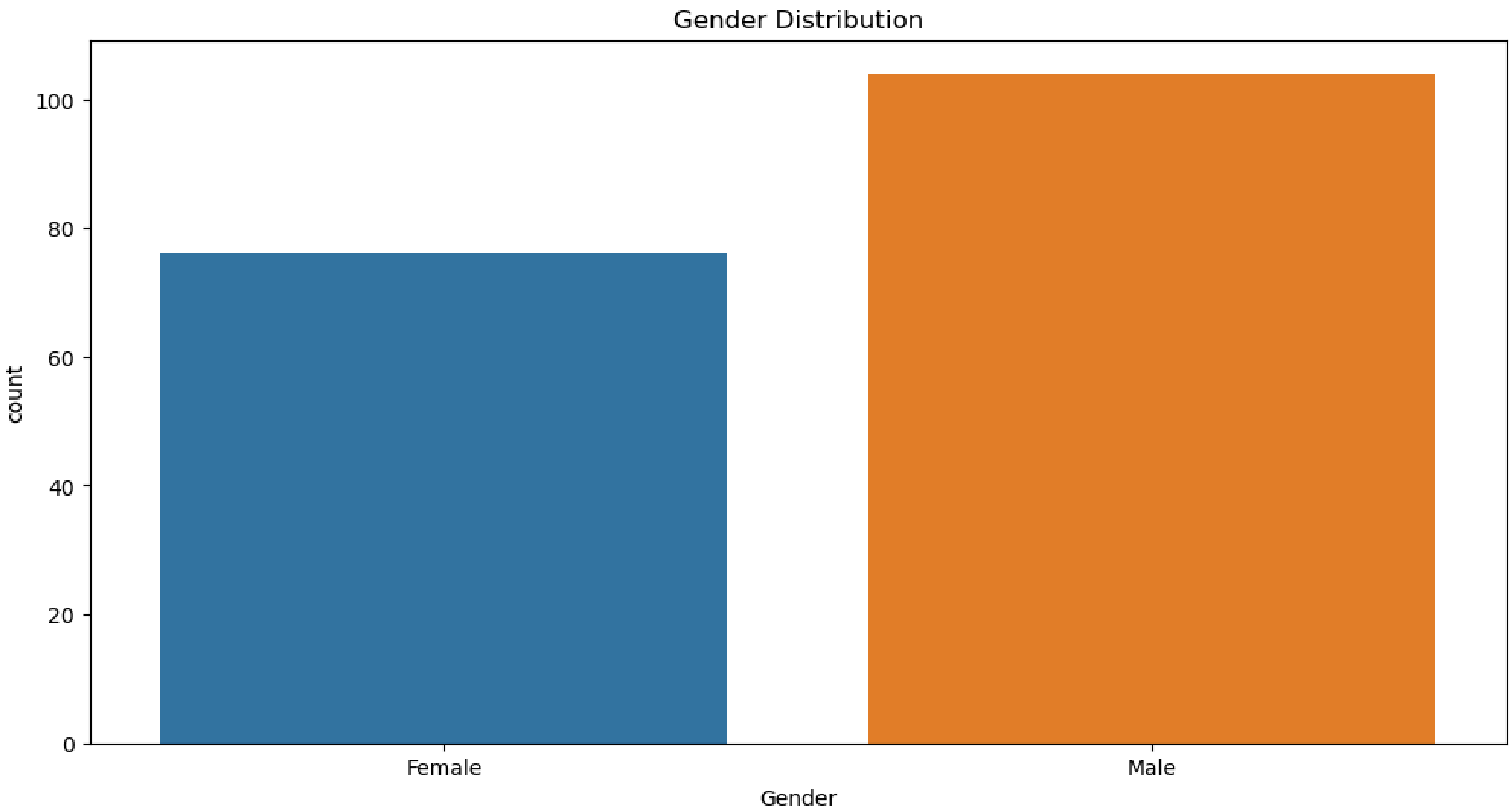
In [58]:

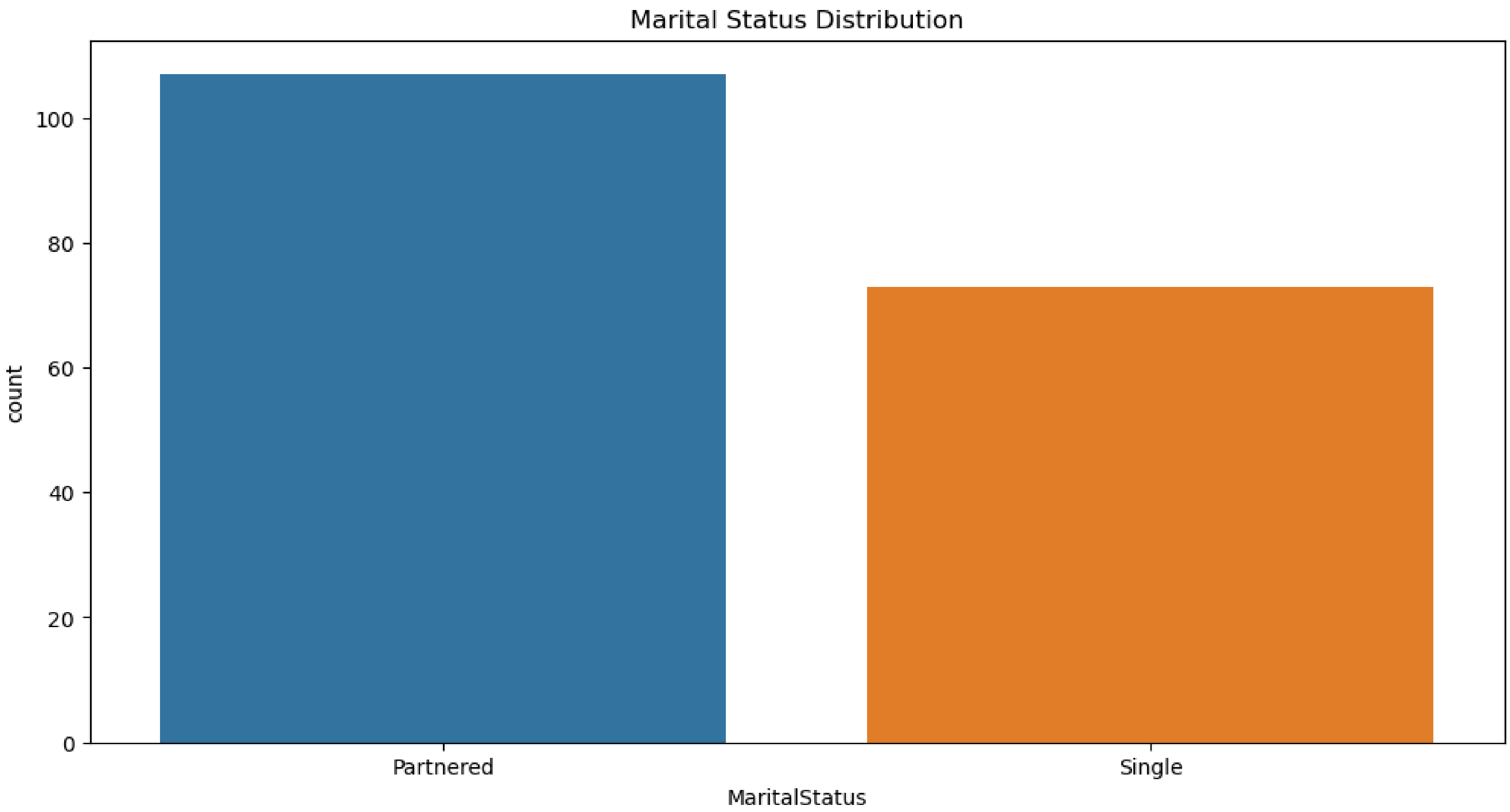
```
# Countplot for Product Purchased
plt.figure(figsize=(12, 6))
sns.countplot(x='Product', data=df)
plt.title('Product Purchased Distribution')
plt.show()

# Countplot for Gender
plt.figure(figsize=(12, 6))
sns.countplot(x='Gender', data=df)
plt.title('Gender Distribution')
plt.show()

# Countplot for Marital Status
plt.figure(figsize=(12, 6))
```

```
sns.countplot(x='MaritalStatus', data=df)
plt.title('Marital Status Distribution')
plt.show()
```

StatementMeta(, 878a4916-1406-44a1-82e3-1c7cbe560263, 60, Finished, Available, Finished)




Gender Distribution
100
80
60
40
20
0
count
Female
Male
Gender

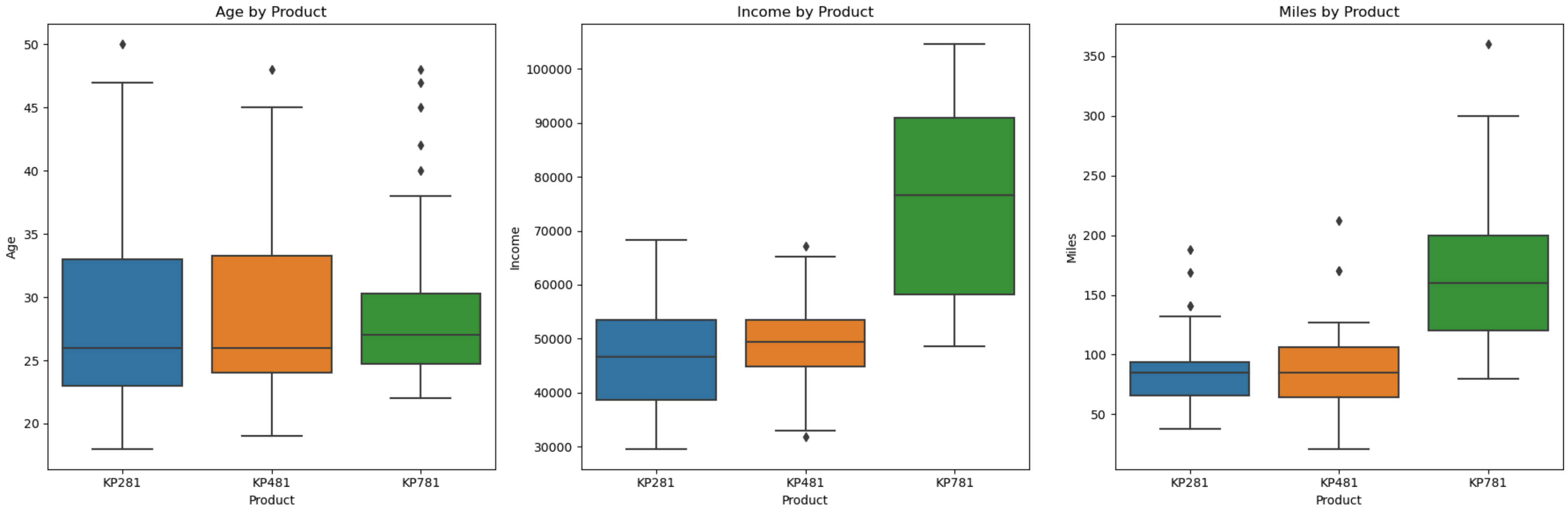


Bivariate Analysis - Boxplots for Product

In [59]:
```
# Boxplots for Age, Income, and Miles by Product
plt.figure(figsize=(18, 6))

# Age by Product
plt.subplot(1, 3, 1)
sns.boxplot(x='Product', y='Age', data=df)
plt.title('Age by Product')
```

```
# Income by Product
plt.subplot(1, 3, 2)
sns.boxplot(x='Product', y='Income', data=df)
plt.title('Income by Product')

# Miles by Product
plt.subplot(1, 3, 3)
sns.boxplot(x='Product', y='Miles', data=df)
plt.title('Miles by Product')

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

StatementMeta(, 878a4916-1406-44a1-82e3-1c7cbe560263, 61, Finished, Available, Finished)



Heatmaps and Pairplots for Correlations

In [60]:
```
# Select only numerical columns for correlation
numerical_df = df.select_dtypes(include=['int64', 'float64'])

# Correlation Heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(numerical_df.corr(), annot=True, cmap='coolwarm', linewidths=0.5)
```

```
plt.title('Correlation Heatmap')
plt.show()
```

StatementMeta(, 878a4916-1406-44a1-82e3-1c7cbe560263, 62, Finished, Available, Finished)

Correlation Heatmap
Age
Education
Usage
Fitness
Income
Miles
1
0.28
0.015
0.061
0.51
0.037
0.4
0.41
0.63
0.31
0.67
0.52
0.76
0.54
0.79
1.0
0.8
0.6
0.4
0.2

Missing Value & Outlier Check

In [61]:
```
##### Missing Value Check:
##### There are no missing values in the dataset
```

```
StatementMeta(, 878a4916-1406-44a1-82e3-1c7cbe560263, 63, Finished, Available, Finished)
```

Outlier Detection:

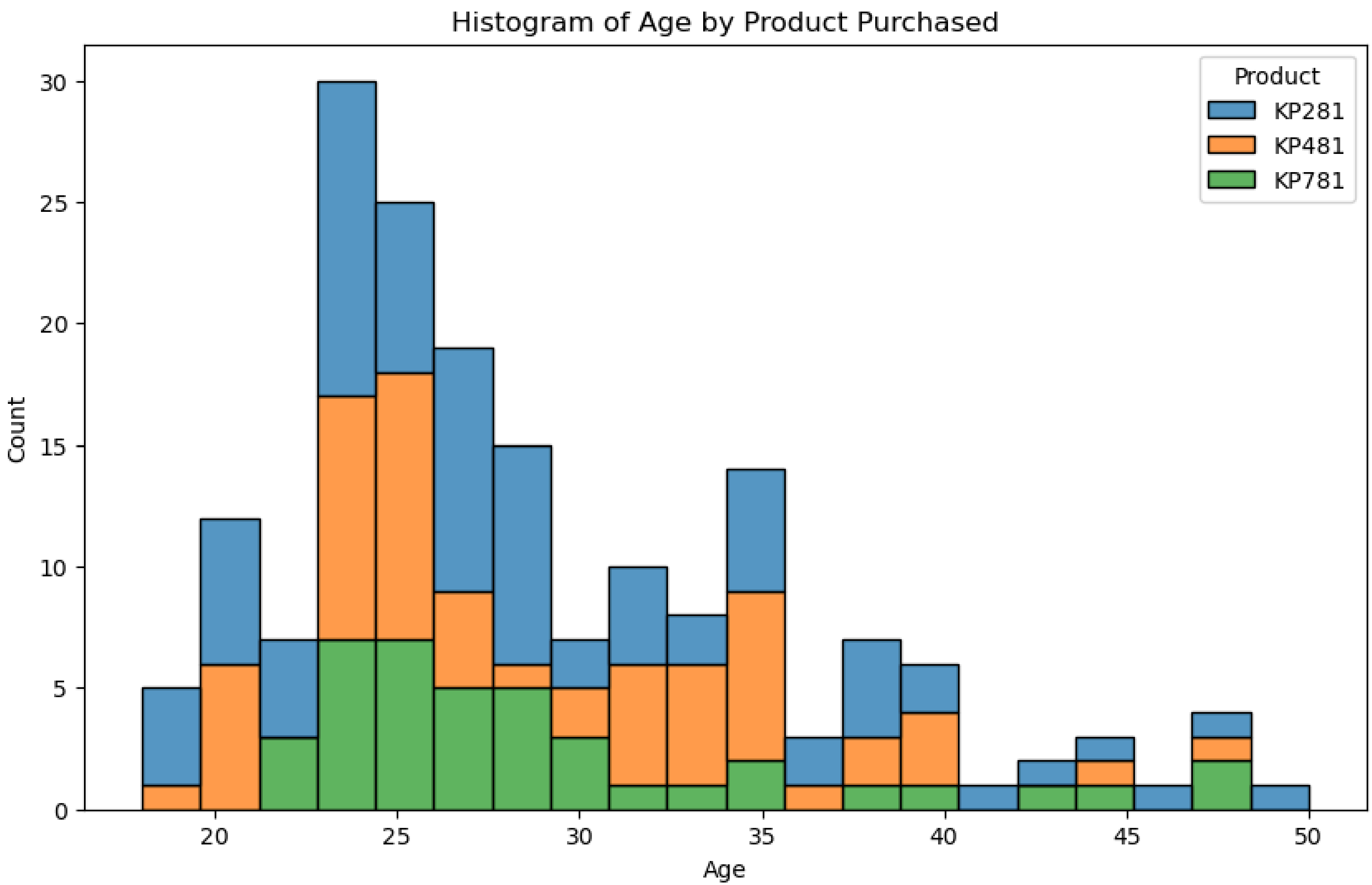
Boxplots: Outliers observed in Age, Income, and Miles.

In [62]:
```
# Difference between mean and median
print(df[['Age', 'Income', 'Usage', 'Miles']].describe())
```

```
StatementMeta(, 878a4916-1406-44a1-82e3-1c7cbe560263, 64, Finished, Available, Finished)
              Age         Income       Usage       Miles
count  180.000000     180.000000  180.000000  180.000000
mean    28.788889   53719.577778    3.455556  103.194444
std      6.943498   16506.684226    1.084797   51.863605
min     18.000000   29562.000000    2.000000   21.000000
25%     24.000000   44058.750000    3.000000   66.000000
50%     26.000000   50596.500000    3.000000   94.000000
75%     33.000000   58668.000000    4.000000  114.750000
max     50.000000  104581.000000    7.000000  360.000000
```

In [63]:
```
# Histogram for Age vs Product Purchased
plt.figure(figsize=(10, 6))
sns.histplot(data=df, x='Age', hue='Product', multiple='stack', bins=20)
plt.title('Histogram of Age by Product Purchased')
plt.show()
```

```
StatementMeta(, 878a4916-1406-44a1-82e3-1c7cbe560263, 65, Finished, Available, Finished)
```



In [64]:
```
# Marginal probabilities
product_counts = df['Product'].value_counts(normalize=True) * 100
print(product_counts)

# Cross-tabulation
product_marital = pd.crosstab(df['Product'], df['MaritalStatus'], normalize='index')
print(product_marital)
```

```
StatementMeta(, 878a4916-1406-44a1-82e3-1c7cbe560263, 66, Finished, Available, Finished)
Product
KP281    44.444444
KP481    33.333333
KP781    22.222222
Name: proportion, dtype: float64
MaritalStatus  Partnered  Single
Product
KP281              0.600   0.400
KP481              0.600   0.400
KP781              0.575   0.425
```

In [65]:
```python
# Conditional probability: Probability of a male customer buying a KP781 treadmill
male_kp781_prob = len(df[(df['Gender'] == 'Male') & (df['Product'] == 'KP781')]) / len(df[df['Gender'] == 'Male'])
print(f"Probability of a male customer buying a KP781 treadmill: {male_kp781_prob:.2%}")
```

```
StatementMeta(, 878a4916-1406-44a1-82e3-1c7cbe560263, 67, Finished, Available, Finished)
Probability of a male customer buying a KP781 treadmill: 31.73%
```

In [66]:
```python
# Conditional probability: Probability of a Partnered customer buying a KP781 treadmill
Partnered_kp781_prob = len(df[(df['MaritalStatus'] == 'Partnered') & (df['Product'] == 'KP781')]) / len(df[df['MaritalStatus']
print(f"Probability of a Partnered customer buying a KP781 treadmill: {Partnered_kp781_prob:.2%}")
```

```
StatementMeta(, 878a4916-1406-44a1-82e3-1c7cbe560263, 68, Finished, Available, Finished)
Probability of a Partnered customer buying a KP781 treadmill: 21.50%
```

Business Insights Based on Non-Graphical and Visual Analysis

Comments on Range of Attributes:

Age ranges from 18 to 50. Income ranges from $29,000 to$105,000. Usage ranges from 2 to 7 times per week. Miles range from 10 to 35 miles per week. Fitness levels range from 1 to 5.

Distribution Comments:

Age: Mostly normally distributed with a peak around 30-40 years. Income: Right-skewed distribution. Usage: Most customers plan to use the treadmill around 3-5 times per week. Miles: Left-skewed distribution, with most customers expecting to run less than 20 miles per week. Fitness: Majority rate their fitness between 3 and 4.

Relationship Comments:

Product vs Age, Income, Miles: Higher-priced products (KP781) are preferred by older, higher-income customers who plan to run more miles