

## ✓ Welcome to Colab!

### Explore the Gemini API

The Gemini API gives you access to Gemini models created by Google DeepMind. Gemini models are built from the ground up to be multimodal, so you can reason seamlessly across text, images, code, and audio.

#### How to get started?

- Go to [Google AI Studio](#) and log in with your Google account.
- [Create an API key](#).
- Use a quickstart for [Python](#), or call the REST API using [curl](#).

#### Discover Gemini's advanced capabilities

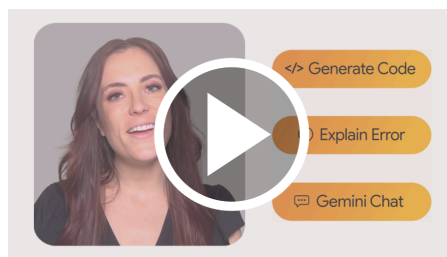
- Play with Gemini [multimodal outputs](#), mixing text and images in an iterative way.
- Discover the [multimodal Live API](#) (demo [here](#)).
- Learn how to [analyze images and detect items in your pictures](#) using Gemini (bonus, there's a [3D version](#) as well!).
- Unlock the power of [Gemini thinking model](#), capable of solving complex task with its inner thoughts.

#### Explore complex use cases

- Use [Gemini grounding capabilities](#) to create a report on a company based on what the model can find on internet.
- Extract [invoices and form data from PDF](#) in a structured way.
- Create [illustrations based on a whole book](#) using Gemini large context window and Imagen.

To learn more, check out the [Gemini cookbook](#) or visit the [Gemini API documentation](#).

Colab now has AI features powered by [Gemini](#). The video below provides information on how to use these features, whether you're new to Python, or a seasoned veteran.



### What is Colab?

Colab, or "Colaboratory", allows you to write and execute Python in your browser, with

- Zero configuration required
- Access to GPUs free of charge
- Easy sharing


Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch [Introduction to Colab](#) or [Colab Features You May Have Missed](#) to learn more, or just get started below!

## ✓ Getting started

The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:

```
seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
```

 86400

To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter". To edit the code, just click the cell and start editing.

Variables that you define in one cell can later be used in other cells:

```
seconds_in_a_week = 7 * seconds_in_a_day
seconds_in_a_week
```

↻ 604800

Colab notebooks allow you to combine **executable code** and **rich text** in a single document, along with **images**, **HTML**, **LaTeX** and more. When you create your own Colab notebooks, they are stored in your Google Drive account. You can easily share your Colab notebooks with co-workers or friends, allowing them to comment on your notebooks or even edit them. To learn more, see [Overview of Colab](#). To create a new Colab notebook you can use the File menu above, or use the following link: [create a new Colab notebook](#).

Colab notebooks are Jupyter notebooks that are hosted by Colab. To learn more about the Jupyter project, see [jupyter.org](#).

## ✓ Data science

With Colab you can harness the full power of popular Python libraries to analyze and visualize data. The code cell below uses **numpy** to generate some random data, and uses **matplotlib** to visualize it. To edit the code, just click the cell and start editing.

You can import your own data into Colab notebooks from your Google Drive account, including from spreadsheets, as well as from Github and many other sources. To learn more about importing data, and how Colab can be used for data science, see the links below under [Working with Data](#).

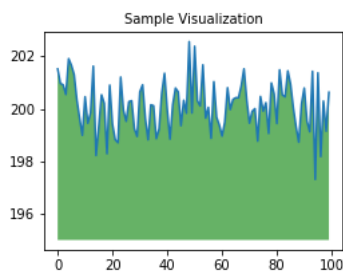
```
import numpy as np
import IPython.display as display
from matplotlib import pyplot as plt
import io
import base64

ys = 200 + np.random.randn(100)
x = [x for x in range(len(ys))]

fig = plt.figure(figsize=(4, 3), facecolor='w')
plt.plot(x, ys, '-')
plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g', alpha=0.6)
plt.title("Sample Visualization", fontsize=10)

data = io.BytesIO()
plt.savefig(data)
image = F"data:image/png;base64,{base64.b64encode(data.getvalue()).decode()}"
alt = "Sample Visualization"
display.display(display.Markdown(F"!!![{alt}]({image})"))
plt.close(fig)
```

↻



Colab notebooks execute code on Google's cloud servers, meaning you can leverage the power of Google hardware, including [GPUs and TPUs](#), regardless of the power of your machine. All you need is a browser.

For example, if you find yourself waiting for **pandas** code to finish running and want to go faster, you can switch to a GPU Runtime and use libraries like [RAPIDS cuDF](#) that provide zero-code-change acceleration.

To learn more about accelerating pandas on Colab, see the [10 minute guide](#) or [US stock market data analysis demo](#).

## ✓ Machine learning

With Colab you can import an image dataset, train an image classifier on it, and evaluate the model, all in just [a few lines of code](#).

Colab is used extensively in the machine learning community with applications including:

- Getting started with TensorFlow
- Developing and training neural networks

- Experimenting with TPUs
- Disseminating AI research
- Creating tutorials

To see sample Colab notebooks that demonstrate machine learning applications, see the [machine learning examples](#) below.

## ✓ More Resources

### Working with Notebooks in Colab

- [Overview of Colab](#)
- [Guide to Markdown](#)
- [Importing libraries and installing dependencies](#)
- [Saving and loading notebooks in GitHub](#)
- [Interactive forms](#)
- [Interactive widgets](#)

### Working with Data

- [Loading data: Drive, Sheets, and Google Cloud Storage](#)
- [Charts: visualizing data](#)
- [Getting started with BigQuery](#)

### Machine Learning

These are a few of the notebooks related to Machine Learning, including Google's online Machine Learning course. See the [full course website](#) for more.

- [Intro to Pandas DataFrame](#)
- [Intro to RAPIDS cuDF to accelerate pandas](#)
- [Getting Started with cuML's accelerator mode](#)
- [Linear regression with tf.keras using synthetic data](#)

### Using Accelerated Hardware

- [TensorFlow with GPUs](#)
- [TPUs in Colab](#)

## ✓ Featured examples

- [Retraining an Image Classifier](#): Build a Keras model on top of a pre-trained image classifier to distinguish flowers.
- [Text Classification](#): Classify IMDB movie reviews as either *positive* or *negative*.
- [Style Transfer](#): Use deep learning to transfer style between images.
- [Multilingual Universal Sentence Encoder Q&A](#): Use a machine learning model to answer questions from the SQuAD dataset.
- [Video Interpolation](#): Predict what happened in a video between the first and the last frame.

### 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.

### 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.

```
import numpy as np          #NECESSARY LIBRARY IMPORTING
import functools as f
```

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv('/content/aerofit_treadmill.csv') #UPLOADING THE DATA
df.head() #WILL SHOW THE TOP 5 ROWS OF THE DATA
```

↗

	Product	Age	Gender	Education	MaritalStatus	Usage	Fitness	Income	Miles	
0	KP281	18	Male	14	Single	3	4	29562	112	il
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](#)

Double-click (or enter) to edit

df.shape will give the no of rows and column. **SO THERE ARE 180 ROWS AND 9 COLUMNS**

df.shape

↗ (180, 9)

df.info()

↗

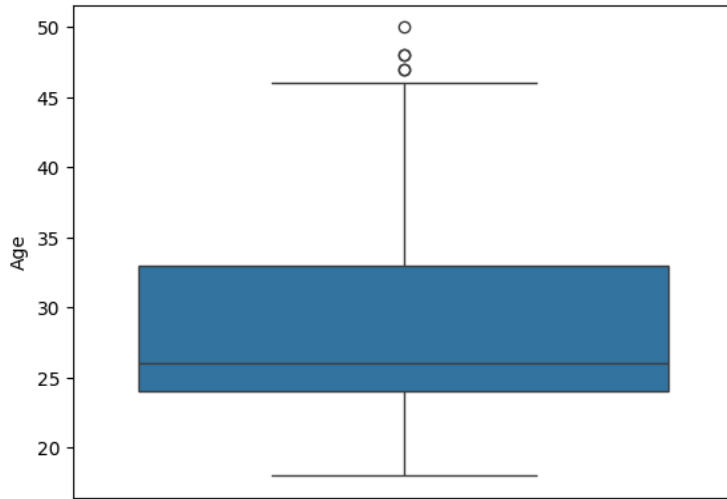
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180 entries, 0 to 179
Data columns (total 9 columns):
#   Column          Non-Null Count  Dtype
---  -
0   Product         180 non-null   object
1   Age             180 non-null   int64
2   Gender          180 non-null   object
3   Education     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

**we can see that there are no null values in any column and most of the data type are int type**

Below is the boxplot of 'Age' column. Looking at the boxplot we can say that there are few outliers. Most of the people buying the product are of age between 24 and 35. **This suggests that people from 24 and 34 are mostly the buyers. so by giving discounts or some other kind of benefits people between the age 34 to 44 should buy the products and become more health concious**

sns.boxplot(df['Age'])

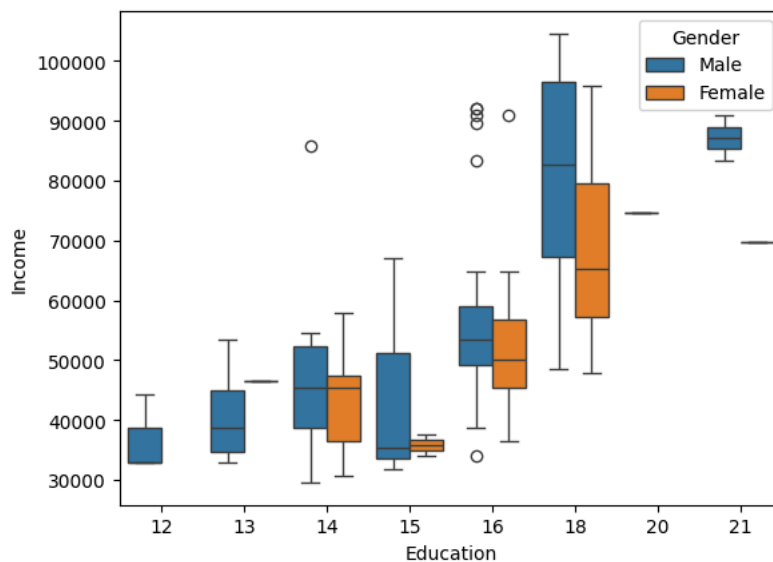
<Axes: ylabel='Age'>



The below boxplot of education suggests following: 1.The buyers having higher education are having higher income. 2.Higher income people are having KP781. 3.Among male and females, males are having higher income. **Suggestions: \*\*The salesman should pitch KP781 if a person is highly educated or have high income**


```
sns.boxplot(x='Education',y='Income',hue='Gender',data=df)
```

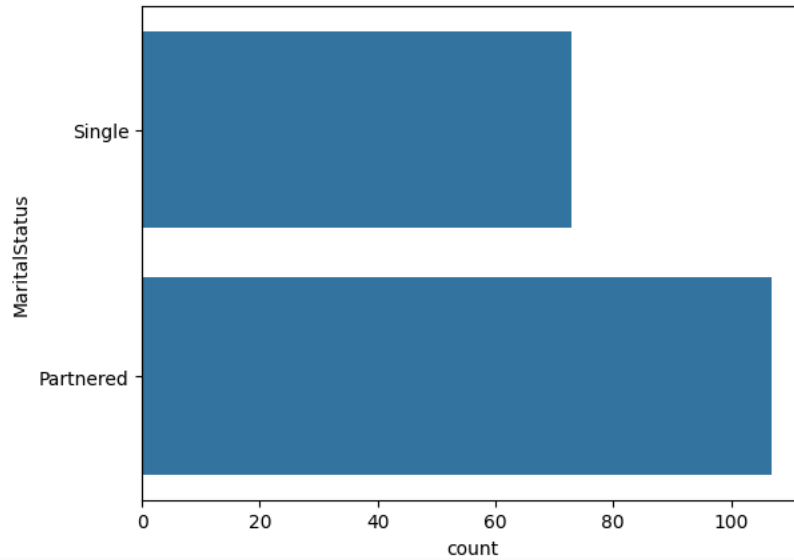
<Axes: xlabel='Education', ylabel='Income'>



Below is the countplot of marital status which shows that most of the buyers are partnered . so strategy should be made to attract more and more partnered people as they are mostly the buyers.


```
sns.countplot(data=df[ 'MaritalStatus' ])
```

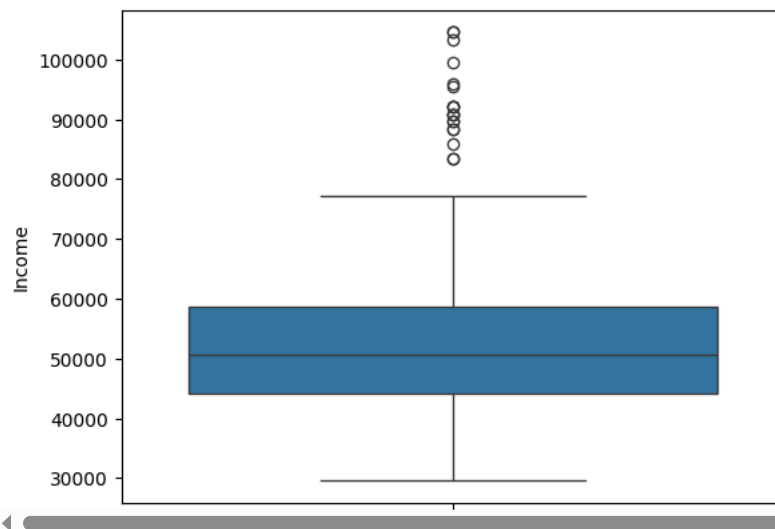
 <Axes: xlabel='count', ylabel='MaritalStatus'>



Below is the boxplot of income of the buyers which shows that the most of the buyers have income between 4200 and 6000. There are few outliers whose income ranges from 8000 and 10500.**the boxplot of income suggests that people who are having high income are interested in buying the treadmill. so proper strategy should be made so to attract high income people**

```
sns.boxplot(df['Income'])
```

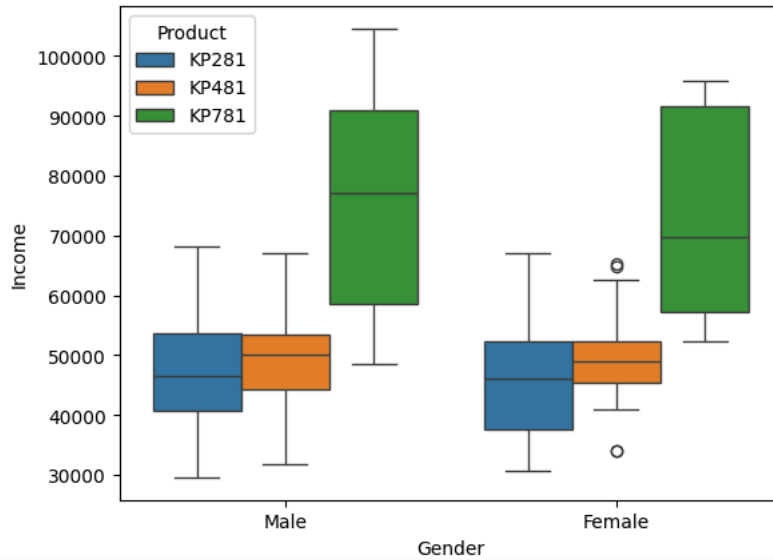
 <Axes: ylabel='Income'>



The below boxplot suggests that people from the gender are buying KP781 if they have high income

```
sns.boxplot(x='Gender', y='Income', hue='Product', data=df)
```

<Axes: xlabel='Gender', ylabel='Income'>



Double-click (or enter) to edit

though we have already seen which of the columns are having outliers but there is one more way to detect outliers. if the mean and median of any column is having high difference then outliers is present in that column but we can see that none of the columns are having high difference which means the outliers are very less and not impacting the data

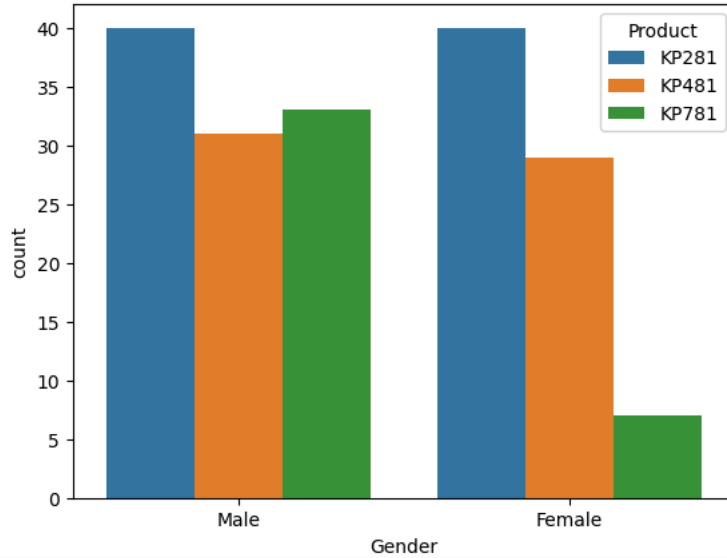
```
df.describe()
```

	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

The below is countplot of Gender and count for each product. **The plot shows that there is hardly any effect of Gender on purchase, but there is difference in product KP781 where female customers are buying this product very less may be because of its high cost** So here some discounts should be given to female customers on the product KP781 for some days. when they have the habit of KP781, they will always buy this

```
sns.countplot(x='Gender', hue='Product', data=df)
```

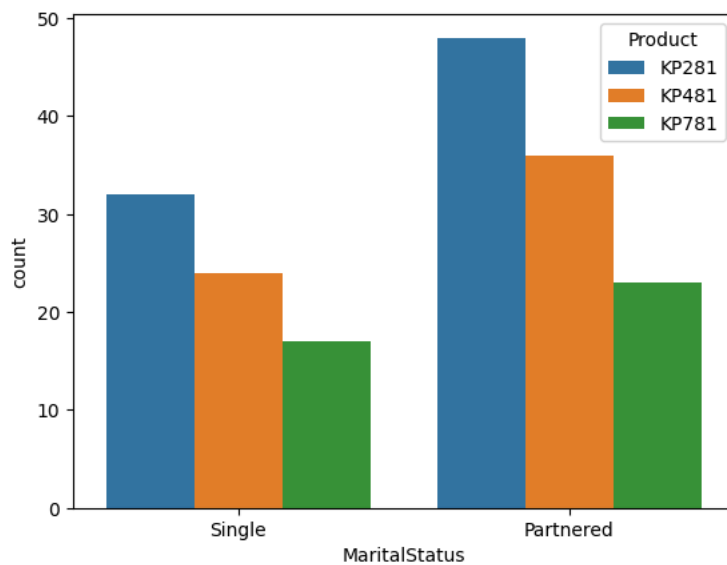
<Axes: xlabel='Gender', ylabel='count'>



By looking at the below graph it can be concluded that marital status is affecting the product buying. the partnered people are buying more products

```
sns.countplot(x='MaritalStatus',hue='Product',data=df)
```

<Axes: xlabel='MaritalStatus', ylabel='count'>



The below table shows us what is the percentage of people buying each product

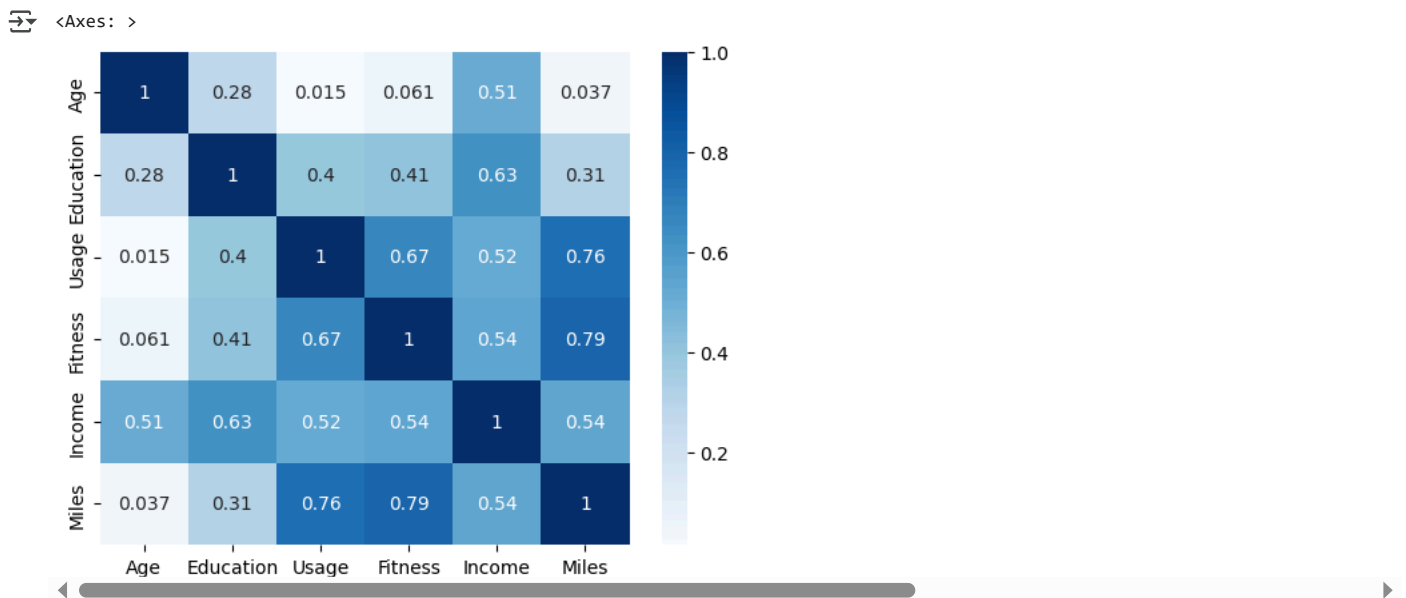
```
a=df['Product'].value_counts().reset_index()
a['percentage_buying']=a['count']/len(df)*100
a
```

	Product	count	percentage_buying
0	KP281	80	44.444444
1	KP481	60	33.333333
2	KP781	40	22.222222

Below is the heat map of the numerical columns of the data the heat map below shows that the usage ,miles and fitness are highly correlated with each other

```
numerical_features = df.select_dtypes(include=np.number).columns
numerical_df = df[numerical_features]
sns.heatmap(numerical_df.corr(), cmap='Blues', annot=True)
```

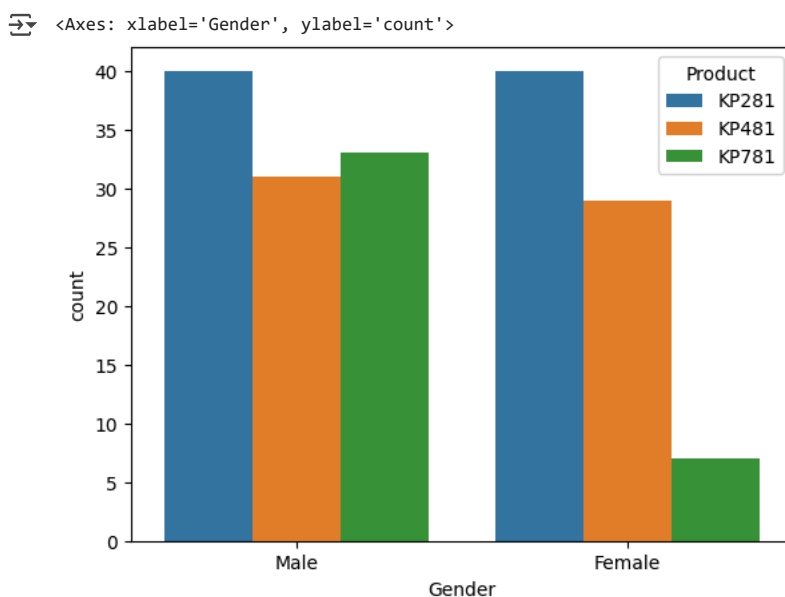




By looking below the count plot we can easily say that the male and female customers have same buying probability for the product KP 281 and also the KP 481 but their buying probability for KP 781 is different. So we can see that the male customers are buying more KP 781 than the female customers

**So we can say that what could be the probability of that a male customer buys a product KP 781 is high. so according to that we can show this product to male customers more likely**

```
sns.countplot(x='Gender', hue='Product', data=df)
```



```
b['count'][b['Gender']=='Male']
```

Below is the data how many male or female customers have bought the different products

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

<Axes: >

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

The below data gives the information:

1. probability of buying any of the products by a male customers is more(0.577) than females
2. the probability of buying KP281,KP481 by any male or female customers are almost same,but probability of buying KP781 by male customers are much more than by females. Suggestions:

*\*The salesman should pitch about KP781 more to males \**

```
pd.crosstab(index=df[ 'Gender' ],columns=df[ 'Product' ],margins=True,normalize=True)
```

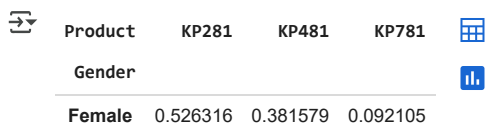


Product	KP281	KP481	KP781	All
Gender				
Female	0.222222	0.161111	0.038889	0.422222
Male	0.222222	0.172222	0.183333	0.577778
All	0.444444	0.333333	0.222222	1.000000

below table gives us the following data:

1. Probability of buying a KP281 if it known that the customer is female ie  $P(KP281|female) = 0.526$
2. Similarly  $P(KP481|female) = 0.381$
3.  $P(KP781|female) = 0.092$ .. etc

```
pd.crosstab(index=df[ 'Gender' ],columns=df[ 'Product' ],margins=True,normalize='index')
```



Product	KP281	KP481	KP781
Gender			
Female	0.526316	0.381579	0.092105