```
############################################  Yossef Nada ##################################################################
# First importing libraries of use

import os
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly
import plotly.express as px



# We will import the data set itself
Imported_data_set=pd.read_csv("train.csv")
# Show the head of the data set
Imported_data_set.head(10)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.250
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.283
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.925
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.100
[illegible]	[illegible]	[illegible]	[illegible]	Allen, Mr. [illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

```
Imported_data_set.info() # just to get an overview of the entire data set , and see the missing data
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

```
# To get the correlation of the data set to understand the relationship between different parameters
Imported_data_set.corr()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005007	-0.035144	0.036847	-0.057527	-0.001652	0.012658
Survived	-0.005007	1.000000	-0.338481	-0.077221	-0.035322	0.081629	0.257307

```
Imported_data_set.isnull().sum() # To get the number of Nulls in each column
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

```
# We will just sort the data ascendingly
total = Imported_data_set.isnull().sum().sort_values(ascending=False)
percent_1 = Imported_data_set.isnull().sum()/Imported_data_set.isnull().count()*100
percent_2 = (round(percent_1, 1)).sort_values(ascending=False)
missing_data = pd.concat([total, percent_2], axis=1, keys=['Total', '%'])
missing_data.head(10)

#################################################################3
# Here we can see that the cabins coloumn is missing more than 77 percent of it's data which means that it might not be of great use to us ir
# Yet through the 'tickets' number and 'pclass' we can find a relation between the cabins and the people staying in them
# The Age coloumn has 177 missing data which will be further assumed by the mean of the age values
```

	Total	%
Cabin	687	77.1
Age	177	19.9
Embarked	2	0.2
PassengerId	0	0.0
Survived	0	0.0
Pclass	0	0.0
Name	0	0.0
Sex	0	0.0
SibSp	0	0.0
Parch	0	0.0

```
Imported_data_set.describe() # just an intial description to get a whole view
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

```
## Check Duplication
Imported_data_set.duplicated()

# The is no duplication in rows
```

```
0      False
1      False
2      False
3      False
4      False
       ...
886    False
887    False
888    False
889    False
890    False
Length: 891, dtype: bool
```

```
# We are going to take the mean of the ages of people and then put that average in the missing data
Imported_data_set.Age=Imported_data_set.Age.fillna(Imported_data_set.Age.median())
```

```
Imported_data_set.info() # Here We see that the Age is now full of data
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          891 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

```
# Here we see that Tickets are not unique which means there are duplicate tickets this might lead to getting a realtionship between the
# Pclass and the lost cabins, similar tickets boarded from the same spot, presumably are from the same class of people and then we might link
# who survived or not
Imported_data_set.nunique()
```

```
PassengerId    891
Survived         2
Pclass           3
Name           891
Sex              2
Age             88
SibSp            7
Parch            7
Ticket         681
Fare           248
Cabin          147
Embarked         3
dtype: int64
```

```
Imported_data_set.dropna(subset=['Embarked'],inplace=True) # remove the rows where we don't know their 'Embarked' details
# Although we might have linked it with the similarity between tickets or fares as we assumed above
```

```
Imported_data_set.info() # just to get an overview of the entire data set
# we see that 2 rows have been dropped
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 889 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  889 non-null    int64
 1   Survived     889 non-null    int64
 2   Pclass       889 non-null    int64
 3   Name         889 non-null    object
 4   Sex          889 non-null    object
 5   Age          889 non-null    float64
 6   SibSp        889 non-null    int64
 7   Parch        889 non-null    int64
 8   Ticket       889 non-null    object
 9   Fare         889 non-null    float64
```

```
 10  Cabin        202 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 90.3+ KB
```

```
Imported_data_set.isnull().sum() # To get the number of Nulls in each column
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         0
dtype: int64
```


```
Imported_data_set.isnull().sum() # To get the number of Nulls in each column
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age              0
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         0
dtype: int64
```


```
Imported_data_set.info() # just to get an overview of the entire data set
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 889 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  889 non-null    int64
 1   Survived     889 non-null    int64
 2   Pclass       889 non-null    int64
 3   Name         889 non-null    object
 4   Sex          889 non-null    object
 5   Age          889 non-null    float64
 6   SibSp        889 non-null    int64
 7   Parch        889 non-null    int64
 8   Ticket       889 non-null    object
 9   Fare         889 non-null    float64
 10  Cabin        202 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 90.3+ KB
```

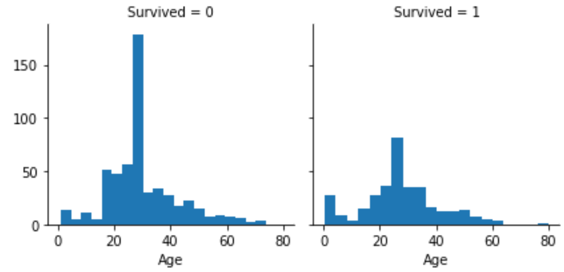
```
Imported_data_set.corr() # Correlation again after tidying up the data set
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.000000	-0.005028	-0.035330	0.031319	-0.057686	-0.001657	0.012703
Survived	-0.005028	1.000000	-0.335549	-0.069822	-0.034040	0.083151	0.255290
Pclass	-0.035330	-0.335549	1.000000	-0.336512	0.081656	0.016824	-0.548193
Age	0.031319	-0.069822	-0.336512	1.000000	-0.232543	-0.171485	0.093707
SibSp	-0.057686	-0.034040	0.081656	-0.232543	1.000000	0.414542	0.160887
Parch	-0.001657	0.083151	0.016824	-0.171485	0.414542	1.000000	0.217532
Fare	0.012703	0.255290	-0.548193	0.093707	0.160887	0.217532	1.000000

```
############################################################
############## Age, Sex and survival ######################
```

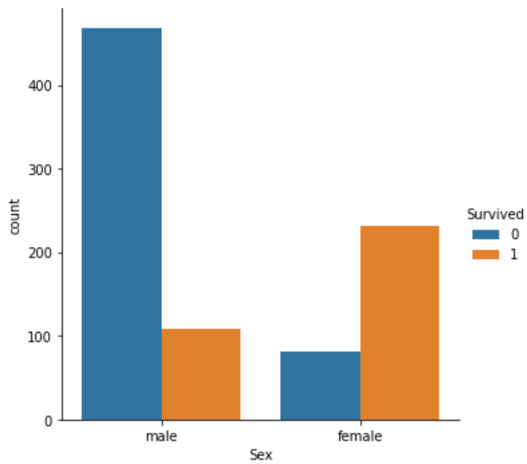
```
############################################################

# We will plot a graph between those who survived and those didnot based on their age differences
g = sns.FacetGrid(Imported_data_set, col='Survived')
g.map(plt.hist, 'Age', bins=20)
```

<seaborn.axisgrid.FacetGrid at 0x7f6cf6cd96a0>



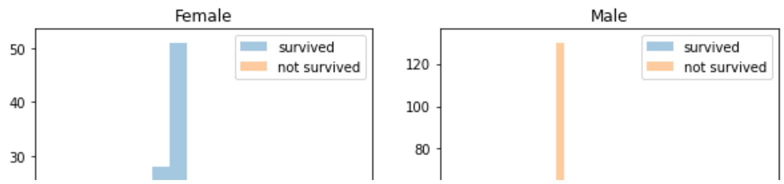

```
# Discussing males and females with those who survived and those who didnot
# and then we will group them all in one graph
sns.catplot(x ="Sex", hue ="Survived",
kind ="count", data = Imported_data_set)
```

<seaborn.axisgrid.FacetGrid at 0x7f6cf6d4e1f0>




```
# Here we will discuss the Age and Sex comparison with those who survived or not

survived = 'survived'
not_survived = 'not survived'
fig, axes = plt.subplots(nrows=1, ncols=2,figsize=(10, 4))
women = Imported_data_set[Imported_data_set['Sex']=='female']
men = Imported_data_set[Imported_data_set['Sex']=='male']
ax = sns.distplot(women[women['Survived']==1].Age.dropna(), bins=18, label = survived, ax = axes[0], kde =False)  # Kernel Density Estimatior
ax = sns.distplot(women[women['Survived']==0].Age.dropna(), bins=40, label = not_survived, ax = axes[0], kde =False)
ax.legend()
ax.set_title('Female')
ax = sns.distplot(men[men['Survived']==1].Age.dropna(), bins=18, label = survived, ax = axes[1], kde = False)
ax = sns.distplot(men[men['Survived']==0].Age.dropna(), bins=40, label = not_survived, ax = axes[1], kde = False)
ax.legend()
_ = ax.set_title('Male')
# You can see that men have a high probability of survival when they are between 18 and 30 years old,
# This might be true for women but not entirely.
# For women the survival chances are higher between 14 and 40.
# For men the probability of survival is very low between the age of 5 and 18, but that isn't true for women.
# Another thing to note is that infants also have a little bit higher probability of survival.
# Since there seem to be certain ages, which have increased odds of survival.
```

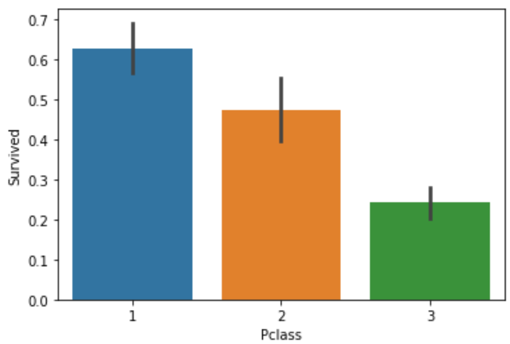
```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a de
  warnings.warn(msg, FutureWarning)
```




```
################################################################
############## Pclass,Fare and survival ###################
################################################################
```

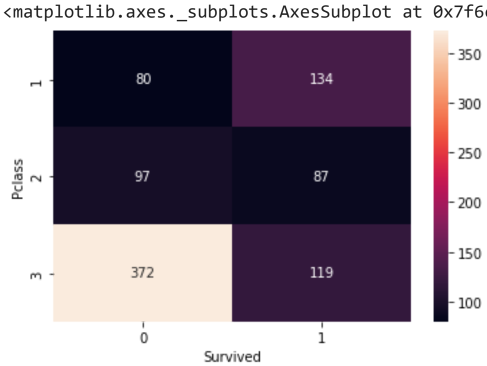
```
# Here we will discuss the classes of the people separatley with those who survived
# we found that class 1 (presumably the wealthiest)are the most people who survived --> Shows patriarchy
sns.barplot(x='Pclass', y='Survived', data=Imported_data_set)

# We assumed that pclass is the top class because they paid for the most expensive tickets
# while class 3 paid for the cheapest tickets
# class 2 was the middle class, there fares was in between
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6cf617f1c0>
```



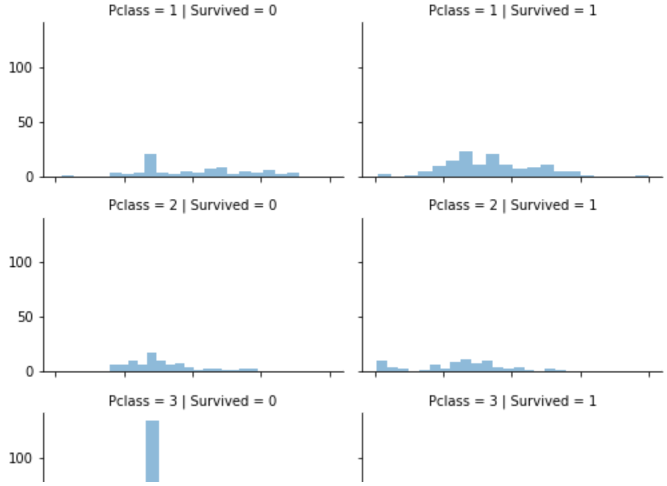

```
# We can also add a heat map of those who survived with discriptive numbers

# Group the dataset by Pclass and Survived and then unstack them
group = Imported_data_set.groupby(['Pclass', 'Survived'])
pclass_survived = group.size().unstack()

# Heatmap - Color encoded 2D representation of data.
sns.heatmap(pclass_survived, annot = True, fmt ="d")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6cf5e5c0a0>
```



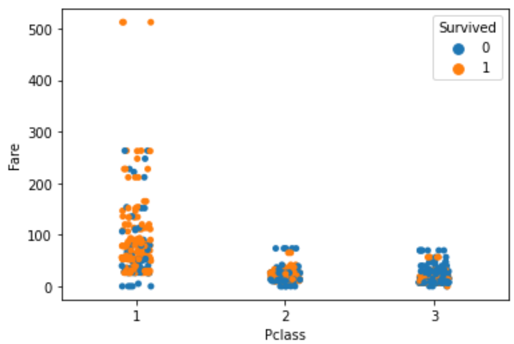

```
grid = sns.FacetGrid(Imported_data_set, col='Survived', row='Pclass', size=2.2, aspect=1.6)
grid.map(plt.hist, 'Age', alpha=.5, bins=20)
grid.add_legend();
#The plot below confirms our assumption about pclass 1, but we can also spot a high probability that a person in pclass 3 will not survive.
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/axisgrid.py:337: UserWarning: The `size` parameter has b
  warnings.warn(msg, UserWarning)
```



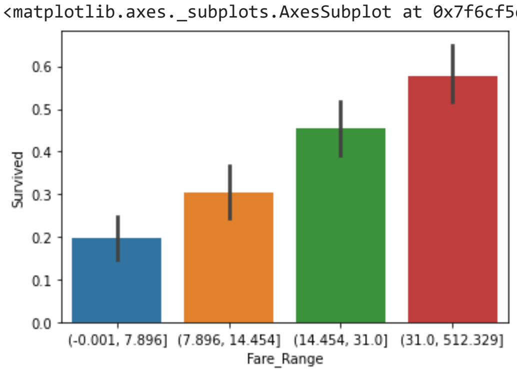

```
# Graphs between Fare and pclass with colors showing those who survived and those who didnot
sns.stripplot(x="Pclass",y="Fare",data=Imported_data_set,hue="Survived")

# From this we see that the pclass those who paid for the highest tickets mostly survived
# but those in class 3, who paid for the cheapest tickets mostly died
# this was probably because of the placement of the cabins in the ships
# Class 3 people were situated in the bottom layer which was probably flooded first
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe405655b50>
```




```
# Another graph to show those who paid the higher fares are the ones who mostly survived and to show the patriarchy in the ship

#Divide Fare into 4 bins
Imported_data_set['Fare_Range'] = pd.qcut(Imported_data_set['Fare'], 4)

# Barplot - Shows approximate values based
# on the height of bars.
sns.barplot(x ='Fare_Range', y ='Survived',
data = Imported_data_set)
```

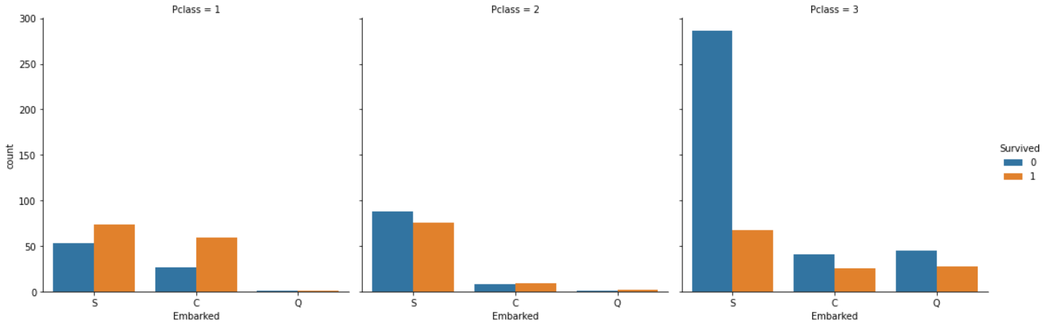
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6cf5d58160>
```




```
##############################################################
############## Sex, Pclass and 'Emabrked' #################
##############################################################
```

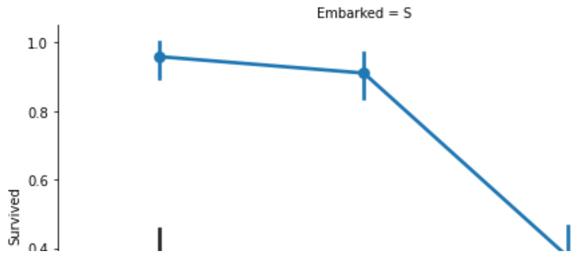
```
# Countplot
sns.catplot(x ='Embarked', hue ='Survived',
kind ='count', col ='Pclass', data = Imported_data_set)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f6cf5cc1b50>
```




```
# Here we will discuss the "Embarked" and "Sex" and "Pclass"

FacetGrid = sns.FacetGrid(Imported_data_set, row='Embarked', size=4.5, aspect=1.6)
FacetGrid.map(sns.pointplot, 'Pclass', 'Survived', 'Sex', palette=None,  order=None, hue_order=None )
FacetGrid.add_legend()


# From the following graphs we see the difference between males and females those who survived and who not and also based on where they boarc
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/axisgrid.py:337: UserWarning: The `size` parameter has b
  warnings.warn(msg, UserWarning)
<seaborn.axisgrid.FacetGrid at 0x7fe405188550>
```

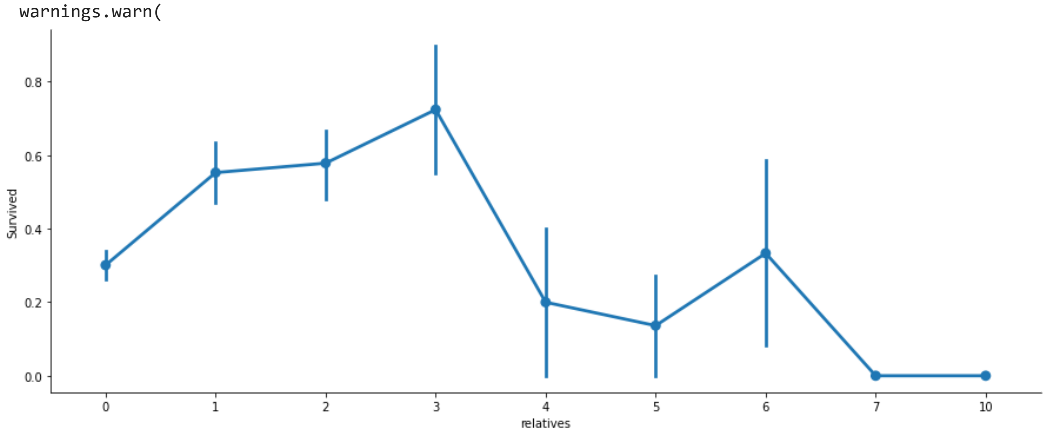



```
############################################################
############## sibsp ,Parch and survival ##################
############################################################

#SibSp and Parch would make more sense as a combined feature, that shows the total number of relatives,
# A person has on the Titanic. I will create it below and also a feature that shows if someone is not alone.
test_df = pd.read_csv("test.csv")
data = [Imported_data_set, test_df]
for dataset in data:
    dataset['relatives'] = dataset['SibSp'] + dataset['Parch']
    dataset.loc[dataset['relatives'] > 0, 'not_alone'] = 0
    dataset.loc[dataset['relatives'] == 0, 'not_alone'] = 1
    dataset['not_alone'] = dataset['not_alone'].astype(int)
Imported_data_set['not_alone'].value_counts()
```

```
1    535
0    354
Name: not_alone, dtype: int64
```

```
axes = sns.factorplot('relatives','Survived',
                      data=Imported_data_set, aspect = 2.5, )


#Here we can see that you had a high probabilty of survival with 1 to 3 realitves,
# but a lower one if you had less than 1 or more than 3 (except for some cases with 6 relatives)
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/categorical.py:3717: UserWarning: The `factorplot` funct
  warnings.warn(msg)
/usr/local/lib/python3.8/dist-packages/seaborn/_decorators.py:36: FutureWarning: Pass the following var
  warnings.warn(
```



✓ 0s completed at 12:29 PM