In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```
df_train = pd.read_csv('train.csv')
```

In [4]:
```
df_train
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

EDA

In [5]:
```
df_train.info()
```

```
<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
```

In [6]:
```
df_train.drop(['PassengerId','Name','Ticket'],axis = 1,inplace = True)
```

In [7]:
```
100*df_train.isna().sum()/df_train.shape[0]## Percentage of Na values in our dataset
```

Out[7]:
```
Survived     0.000000
Pclass       0.000000
Sex          0.000000
Age         19.865320
SibSp        0.000000
Parch        0.000000
Fare         0.000000
Cabin       77.104377
Embarked     0.224467
dtype: float64
```

In [8]:
```
df_train.head()
```

Out[8]:

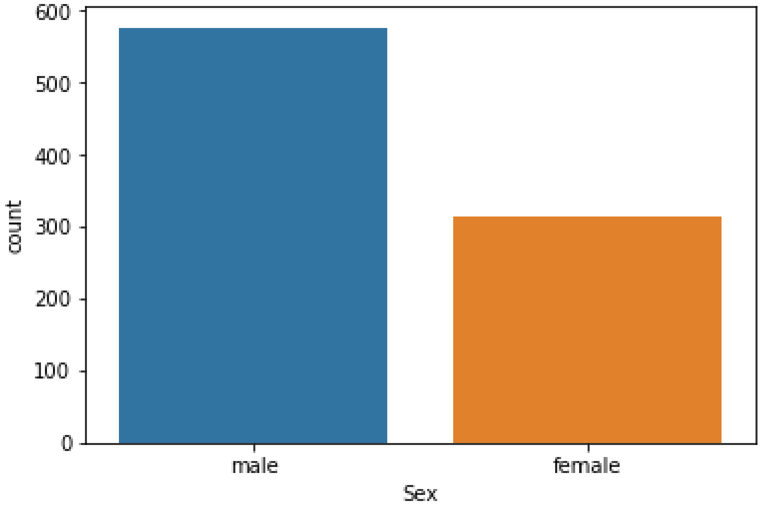
	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked
0	0	3	male	22.0	1	0	7.2500	NaN	S
1	1	1	female	38.0	1	0	71.2833	C85	C
2	1	3	female	26.0	0	0	7.9250	NaN	S
3	1	1	female	35.0	1	0	53.1000	C123	S
4	0	3	male	35.0	0	0	8.0500	NaN	S

Questions to ask:

1. Analyse the distribution in X variable/Who were the passengers
2. Analyse the distribution in the Y variable/Understand the distribution in the Survived column
3. Analyse the relation between X and Y/ Find out which attributes are affecting the Y variable

In [9]:
```
## Distribution in sex column
sns.countplot(x = 'Sex',data = df_train)
```

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x26e9558bdc8>

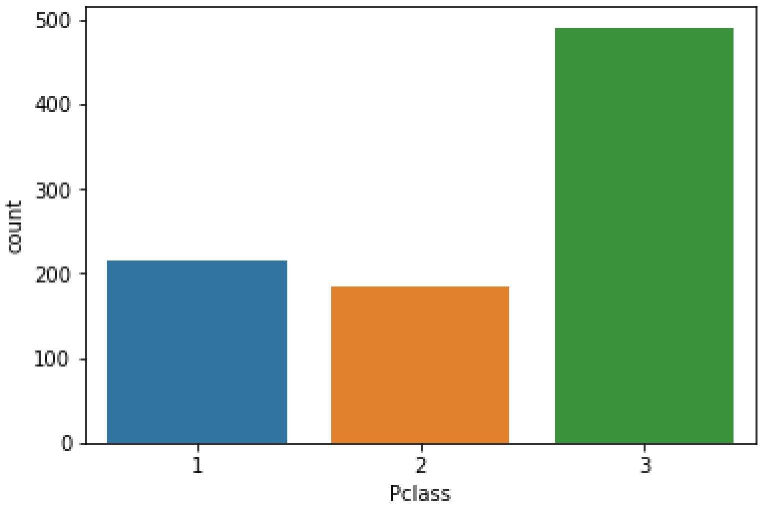


In [10]:
```
df_train['Sex'].value_counts()/df_train.shape[0]
```

Out[10]:
```
male      0.647587
female    0.352413
Name: Sex, dtype: float64
```

In [11]:
```
## Distribution in Pclass column
sns.countplot(x = 'Pclass',data = df_train)
```

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x26e956d67c8>

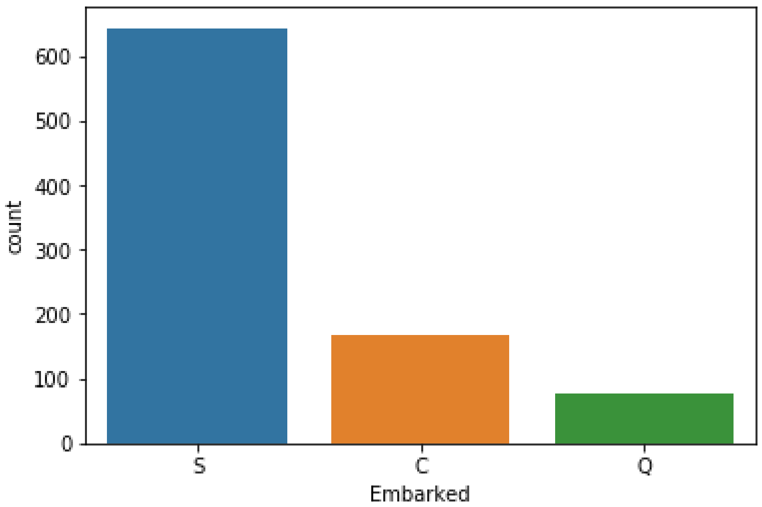


In [12]:
```
df_train['Pclass'].value_counts()/df_train.shape[0]
```

Out[12]:
```
3    0.551066
1    0.242424
2    0.206510
Name: Pclass, dtype: float64
```

In [13]:
```
## Distribution in Embarked column
sns.countplot(x = 'Embarked',data = df_train)
```

Out[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x26e9573a788>
```

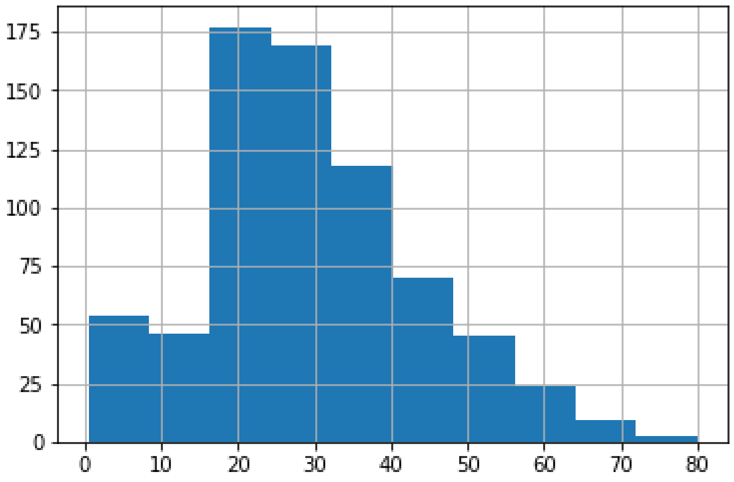


In [14]:
```
df_train['Embarked'].value_counts()/df_train.shape[0]
```

Out[14]:
```
S    0.722783
C    0.188552
Q    0.086420
Name: Embarked, dtype: float64
```

In [15]:
```
df_train['Age'].hist()
```

Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x26e95c01148>

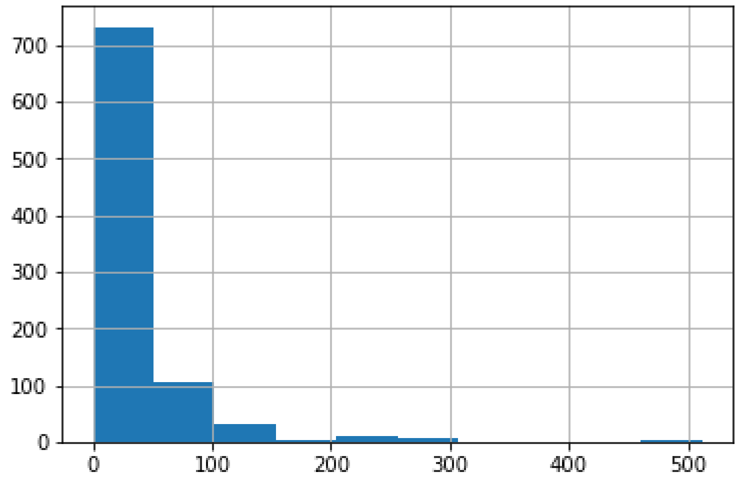


In [16]:
```
df_train['Age'].describe()
```

Out[16]:
```
count    714.000000
mean      29.699118
std       14.526497
min        0.420000
25%       20.125000
50%       28.000000
75%       38.000000
max       80.000000
Name: Age, dtype: float64
```

In [17]:
```
df_train['Fare'].hist()
```

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x26e95a2de08>



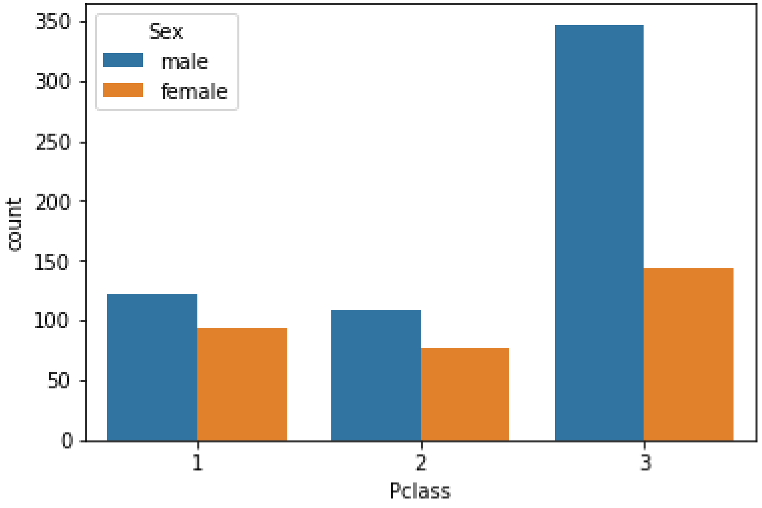
In [18]:
```
df_train['Fare'].describe()
```

Out[18]:
```
count    891.000000
mean      32.204208
std       49.693429
min        0.000000
25%        7.910400
50%       14.454200
75%       31.000000
max      512.329200
Name: Fare, dtype: float64
```

Multivariate Analysis

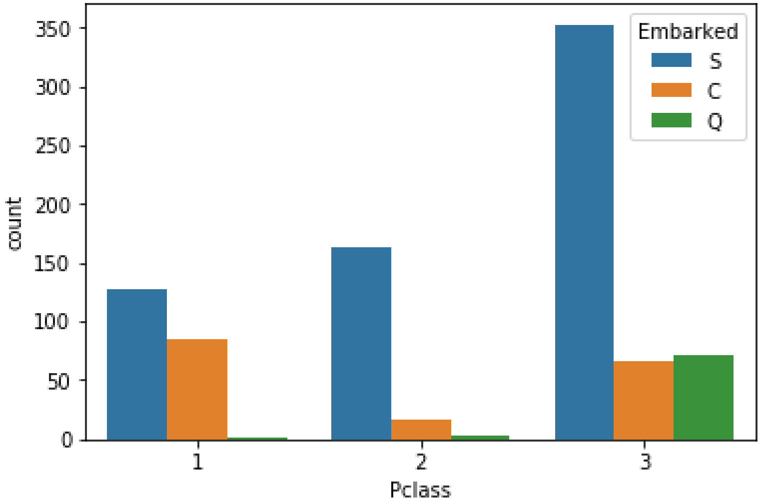
In [19]:
```
sns.countplot(x = 'Pclass',data = df_train,hue = 'Sex')
## homework: Find the numbers eg in plcass 3 what is the percentage of males vs females
```

Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x26e95b2fb48>



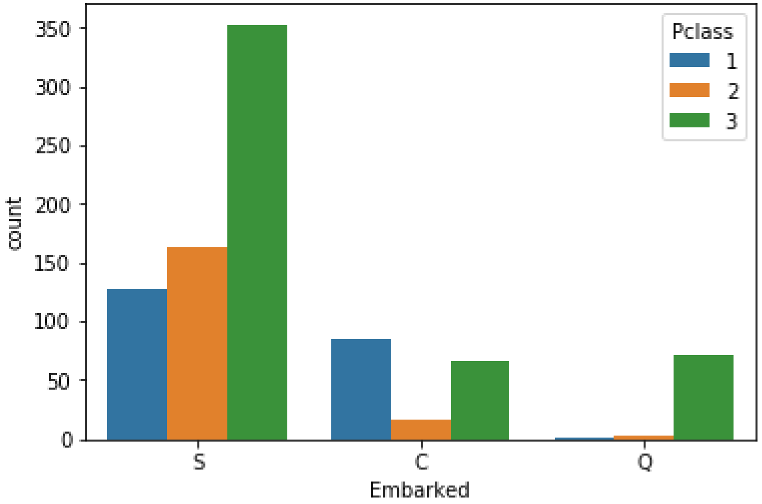
In [20]:
```
sns.countplot(x = 'Pclass',data = df_train,hue = 'Embarked')
```

Out[20]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x26e95c8f148>
```



In [21]:
```
sns.countplot(x = 'Embarked',data = df_train,hue = 'Pclass')
```

Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x26e95ce7d48>



In [22]:
```
df_train['Cabin'].isna().sum()/df_train.shape[0]
## Depending on your domain and column importance if you have more than 25-30% na values in any particular column yo
```

Out[22]: 0.7710437710437711

In [23]:
```
for i in df_train['Embarked'].dropna().unique():
    print(f"Passangers Embarked from Port {i} : the distribution is \n {100*df_train[df_train['Embarked']==i]['Pclas
```

```
Passangers Embarked from Port S : the distribution is
 3    39.618406
2    18.406285
1    14.253648
Name: Pclass, dtype: float64
Passangers Embarked from Port C : the distribution is
 1    9.539843
3    7.407407
2    1.907969
Name: Pclass, dtype: float64
Passangers Embarked from Port Q : the distribution is
 3    8.080808
2    0.336700
1    0.224467
Name: Pclass, dtype: float64
```

In [24]:
```
df_train['Cabin']
```

Out[24]:
```
0       NaN
1       C85
2       NaN
3      C123
4       NaN
       ...
886     NaN
887     B42
888     NaN
889    C148
890     NaN
Name: Cabin, Length: 891, dtype: object
```

In [25]:
```
levels = []

levels = [i[0] for i in df_train['Cabin'].dropna()]
```

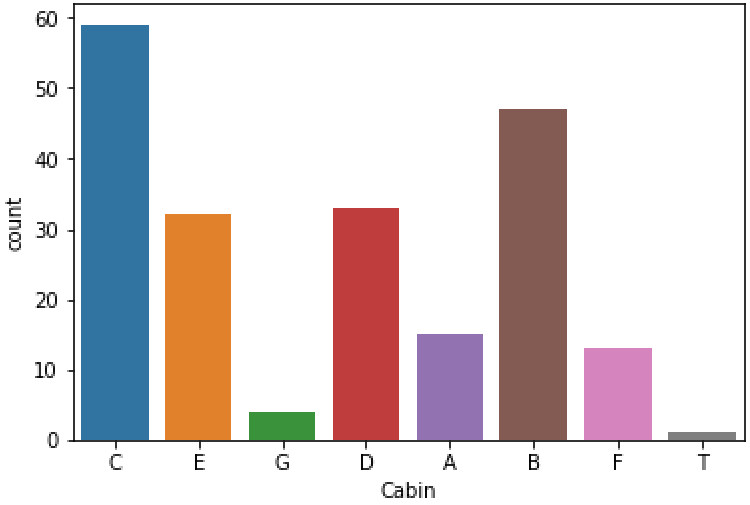
In [26]:
```
levels
```

```
'E',
'E',
'C',
'C',
'C',
'F',
'C',
'E',
'E',
'B',
'B',
'D',
'C',
'B',
'B',
'D',
'E',
'B',
'B',
'D',
```

In [27]:
```
cabin = pd.DataFrame(levels,columns = ['Cabin'])
```

In [28]:
```
sns.countplot(x = 'Cabin',data = cabin)
```

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x26e95d94548>



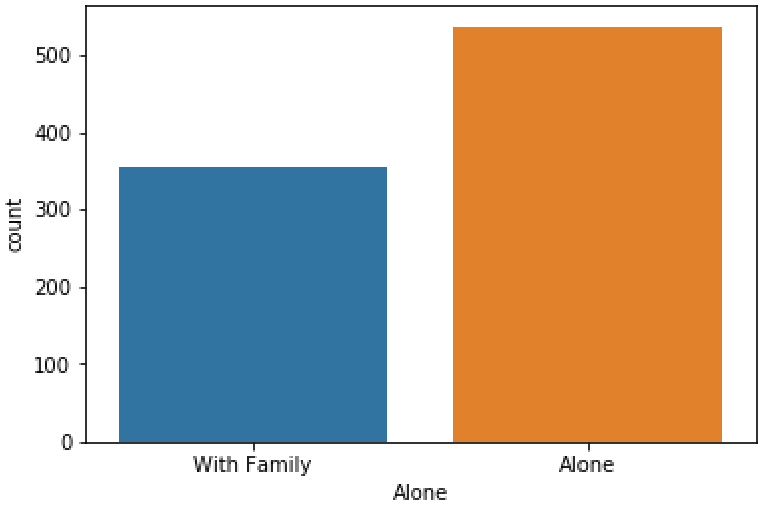
In [29]:
```
df_train['Alone'] = df_train['SibSp'] + df_train['Parch']
```

In [30]:
```
df_train['Alone'].loc[df_train['Alone']>0] = 'With Family'
df_train['Alone'].loc[df_train['Alone']==0] = 'Alone'
```

```
C:\Users\yashm\anaconda3\lib\site-packages\pandas\core\indexing.py:1732: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a
-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-co
py)
  self._setitem_single_block(indexer, value, name)
```

In [31]:
```
sns.countplot(x = 'Alone',data = df_train)
```

Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x26e95e23708>

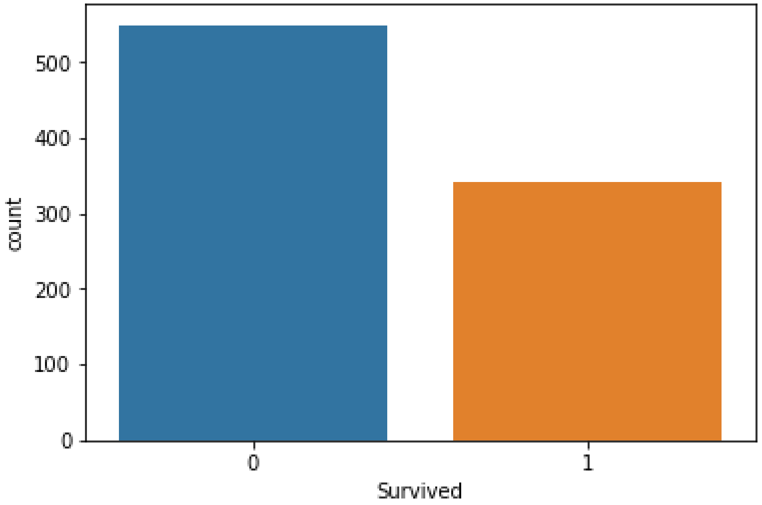


Analyse the distribution in the Y variable/Understand the distribution in the Survived column

In [32]:

```
sns.countplot(x = 'Survived',data = df_train)
```

Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x26e95e671c8>



In [33]:

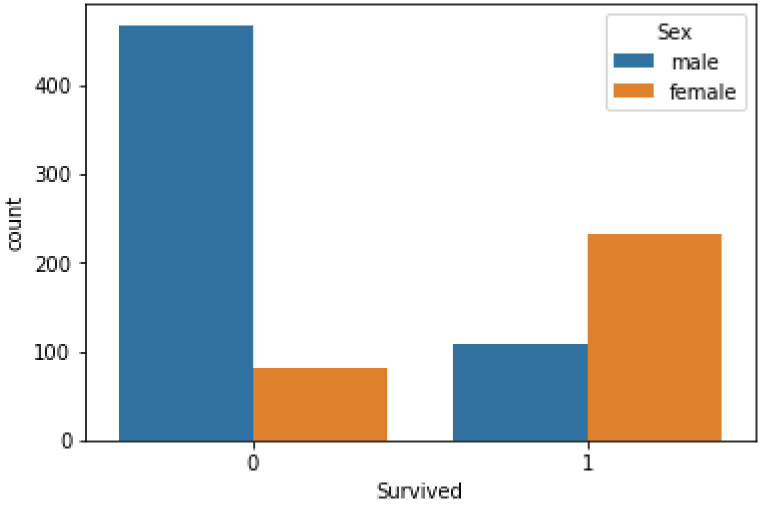
```
df_train['Survived'].value_counts()/df_train.shape[0]
```

Out[33]:

```
0    0.616162
1    0.383838
Name: Survived, dtype: float64
```

In [34]:
```
sns.countplot(x = 'Survived',data = df_train,hue = 'Sex')
```

Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x26e95ebefc8>



In [35]:
```
df_train
```

Out[35]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked	Alone
0	0	3	male	22.0	1	0	7.2500	NaN	S	With Family
1	1	1	female	38.0	1	0	71.2833	C85	C	With Family
2	1	3	female	26.0	0	0	7.9250	NaN	S	Alone
3	1	1	female	35.0	1	0	53.1000	C123	S	With Family
4	0	3	male	35.0	0	0	8.0500	NaN	S	Alone
...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13.0000	NaN	S	Alone
887	1	1	female	19.0	0	0	30.0000	B42	S	Alone
888	0	3	female	NaN	1	2	23.4500	NaN	S	With Family
889	1	1	male	26.0	0	0	30.0000	C148	C	Alone
890	0	3	male	32.0	0	0	7.7500	NaN	Q	Alone

891 rows × 10 columns

In [69]:
```
def Person(Passenger):
    Sex,Age = Passenger
    if Age < 18:
        return "child"
    else:
        return Sex


```

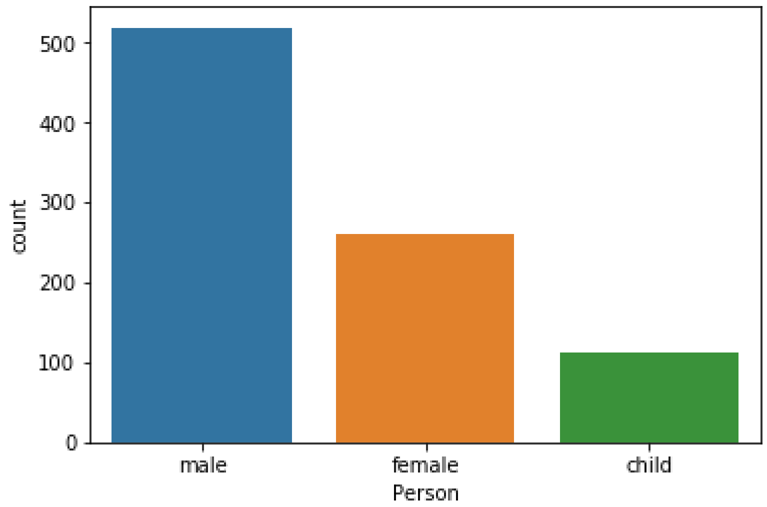
In [70]:
```
df_train['Person'] = df_train[['Sex','Age']].apply(Person,axis = 1)
```

In [71]:
```
df_train['Person']
```

Out[71]:
```
0        male
1      female
2      female
3      female
4        male
        ...  
886      male
887    female
888    female
889      male
890      male
Name: Person, Length: 891, dtype: object
```

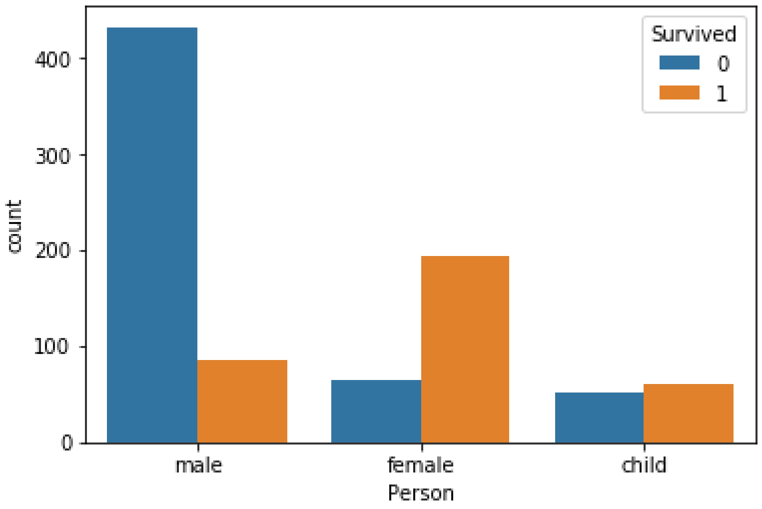
In [72]:
```
sns.countplot(x = 'Person',data = df_train)
```

Out[72]: <matplotlib.axes._subplots.AxesSubplot at 0x26e98cf2148>



In [73]:
```
sns.countplot(x = 'Person',data = df_train,hue = 'Survived')
```

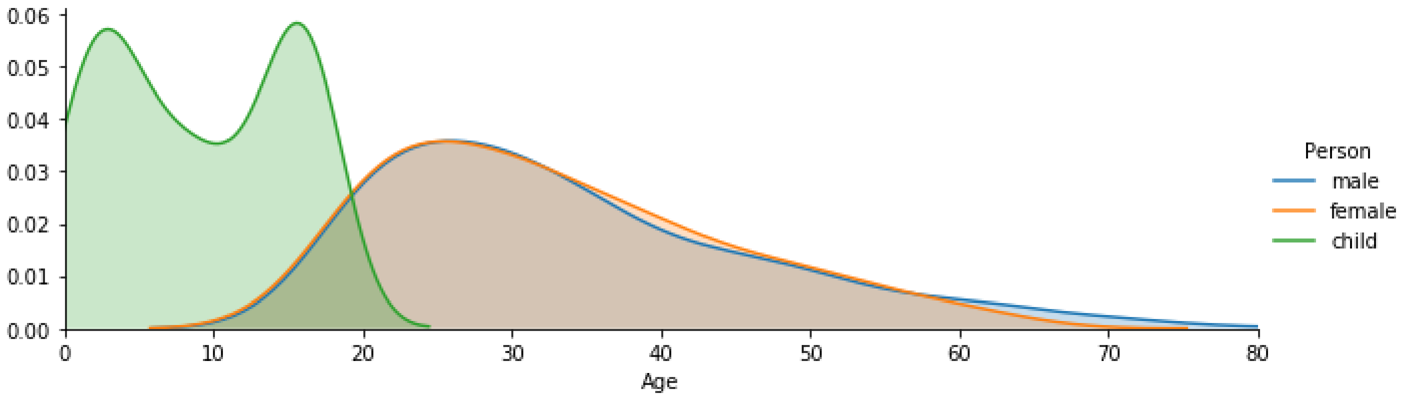
Out[73]: <matplotlib.axes._subplots.AxesSubplot at 0x26e9a323808>



Numerical variables vs categoricals

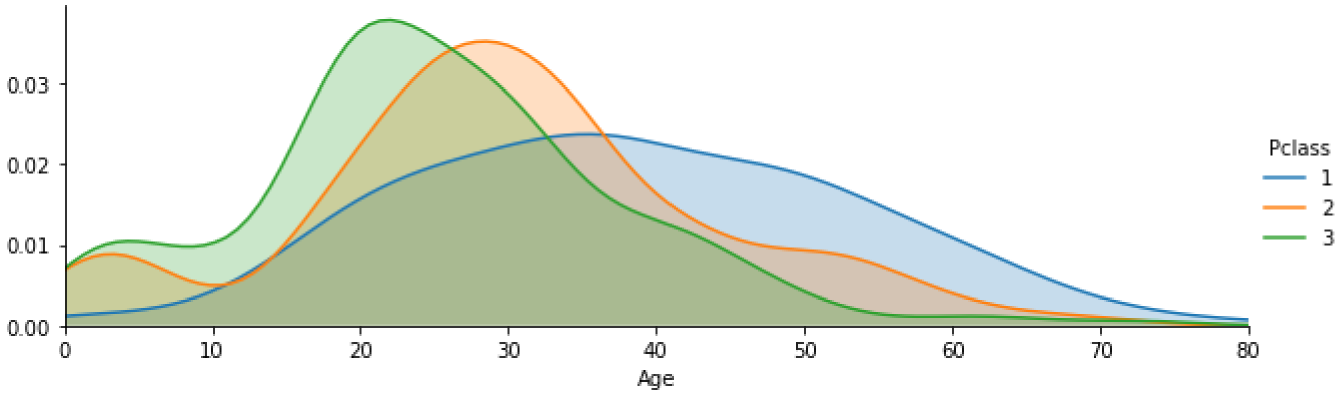
In [74]:
```
fig  = sns.FacetGrid(df_train,hue = 'Person',aspect = 3)
fig.map(sns.kdeplot,'Age',shade = True)

oldest = df_train['Age'].max()

fig.set(xlim = (0,oldest))

fig.add_legend()
```

Out[74]: <seaborn.axisgrid.FacetGrid at 0x26e9a38aa48>



In [75]:
```
fig  = sns.FacetGrid(df_train,hue = 'Pclass',aspect = 3)
fig.map(sns.kdeplot,'Age',shade = True)

oldest = df_train['Age'].max()

fig.set(xlim = (0,oldest))

fig.add_legend()
```

Out[75]: <seaborn.axisgrid.FacetGrid at 0x26e9a3df348>

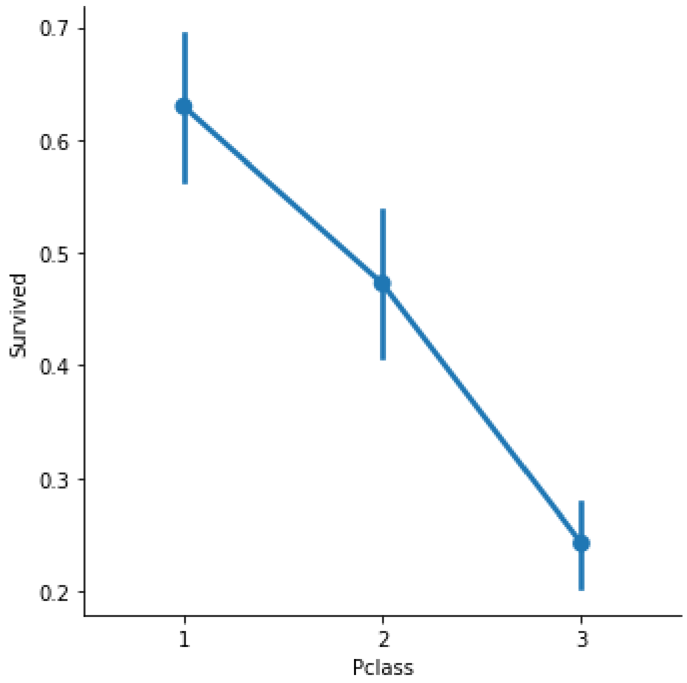


Conclusive graphs/Analyse the relation between X and Y/ Find out which attributes are affecting the Y variable

In [77]:

```
# cat vs cat

sns.factorplot(x = 'Pclass',y = 'Survived',data = df_train)
```

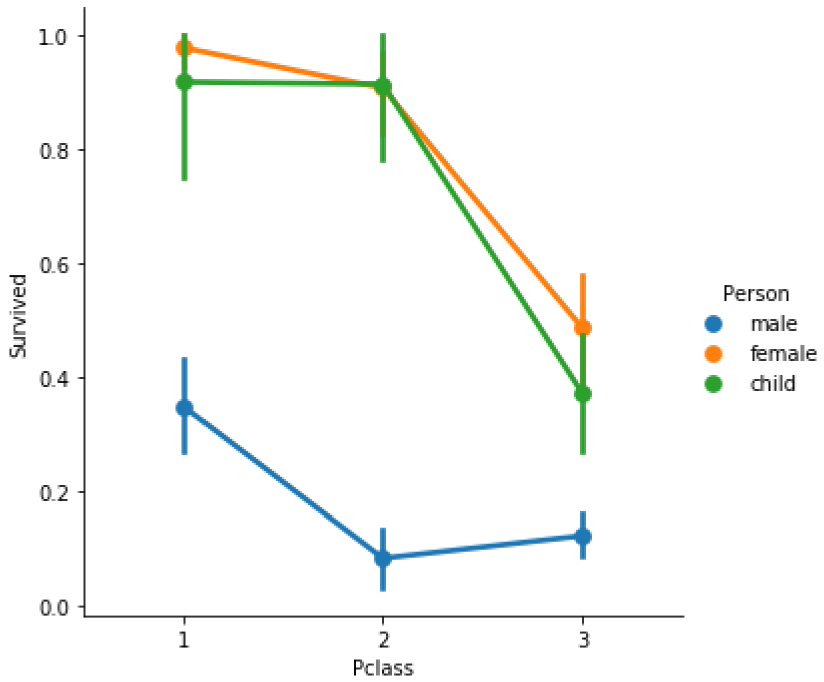
Out[77]: <seaborn.axisgrid.FacetGrid at 0x26e9906a288>



In [78]:
```
sns.factorplot(x = 'Pclass',y = 'Survived',data = df_train,hue = 'Person')
```

```
C:\Users\yashm\anaconda3\lib\site-packages\seaborn\categorical.py:3669: UserWarning: The `factorplot` function has been renamed to `catplot`. The original name will be removed in a future release. Please update your code. Note that the default `kind` in `factorplot` (`'point'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
```

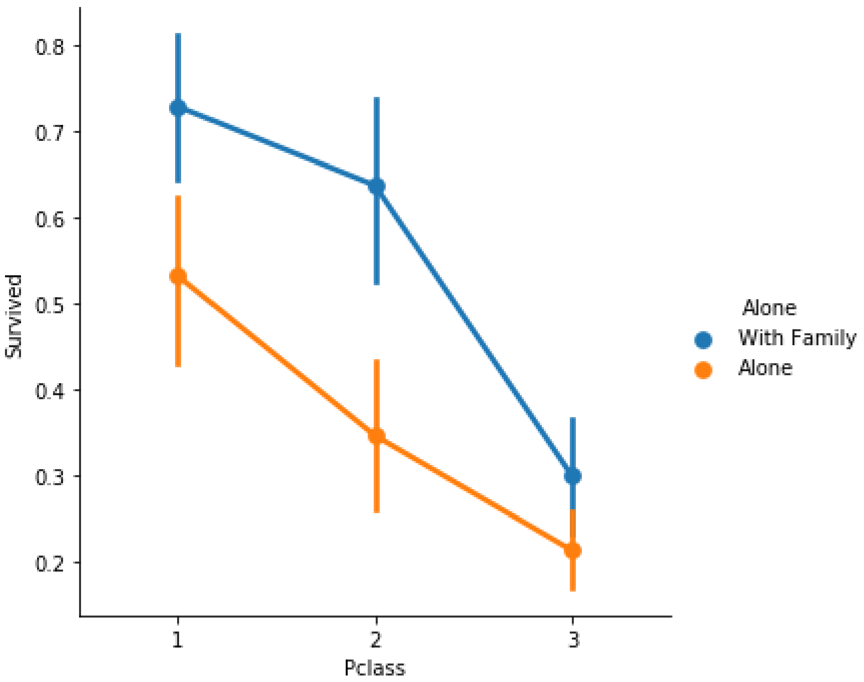
Out[78]: <seaborn.axisgrid.FacetGrid at 0x26e9a59dc08>



In [79]:
```
sns.factorplot(x = 'Pclass',y = 'Survived',data = df_train,hue = 'Alone')
```

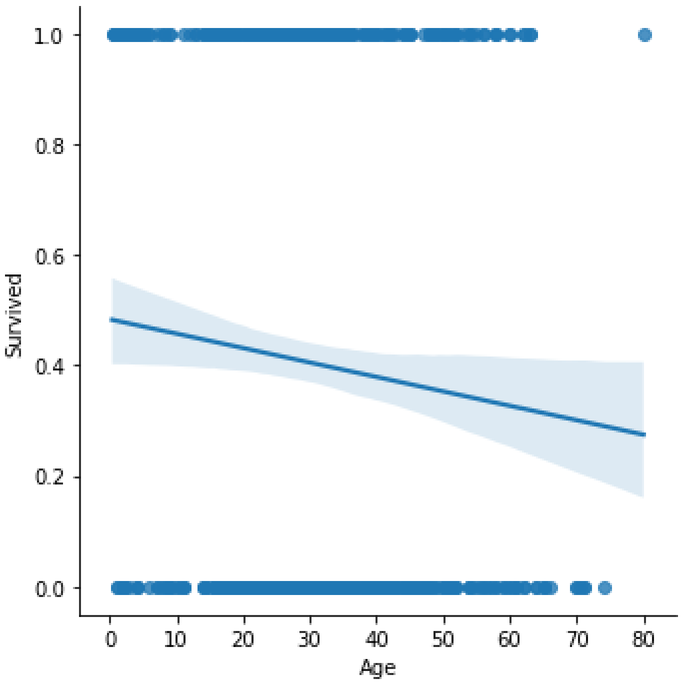
```
C:\Users\yashm\anaconda3\lib\site-packages\seaborn\categorical.py:3669: UserWarning: The `factorplot` function has been renamed to `catplot`. The original name will be removed in a future release. Please update your code. Note that the default `kind` in `factorplot` (`'point'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
```

Out[79]: <seaborn.axisgrid.FacetGrid at 0x26e9a572f48>



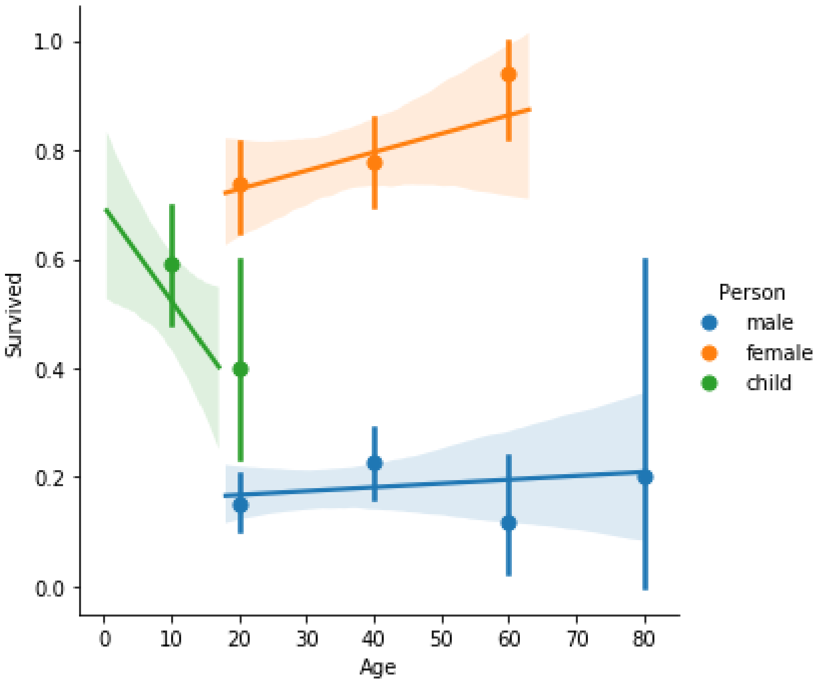
In [80]:
```
# numerical vs cat/numerical vs numerical

sns.lmplot('Age','Survived',df_train)
```

Out[80]: <seaborn.axisgrid.FacetGrid at 0x26e9a5a9948>



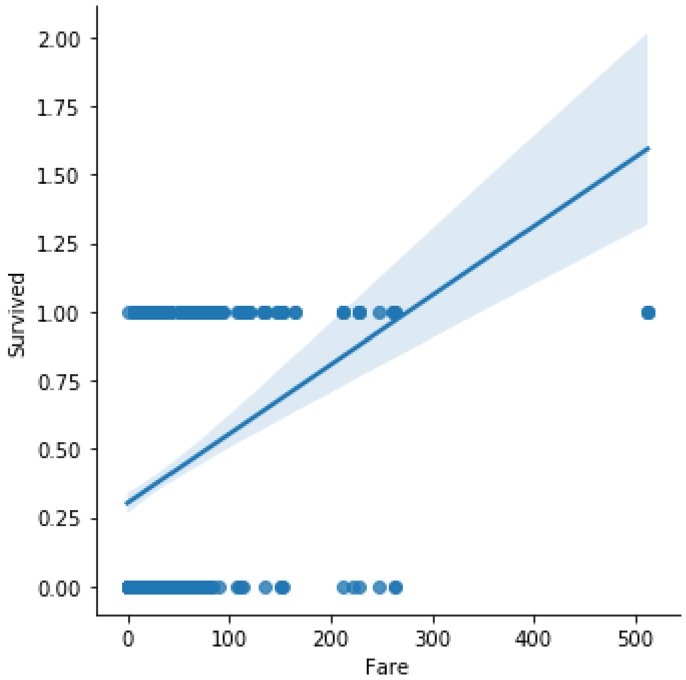
In [83]:
```
sns.lmplot('Age','Survived',df_train,hue = 'Person',x_bins = [10,20,40,60,80])
```

Out[83]: <seaborn.axisgrid.FacetGrid at 0x26e9b795b08>



In [84]:
```
sns.lmplot('Fare','Survived',df_train)
```

Out[84]: <seaborn.axisgrid.FacetGrid at 0x26e9b81e608>



Homework

Write down the inference that we have discussed and if you find anymore for each of these graphs

Model Building

Data Cleaning

In [85]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
```

In [86]:
```
df_train.head()
```

Out[86]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Cabin	Embarked	Alone	Person
0	0	3	male	22.0	1	0	7.2500	NaN	S	With Family	male
1	1	1	female	38.0	1	0	71.2833	C85	C	With Family	female
2	1	3	female	26.0	0	0	7.9250	NaN	S	Alone	female
3	1	1	female	35.0	1	0	53.1000	C123	S	With Family	female
4	0	3	male	35.0	0	0	8.0500	NaN	S	Alone	male

In [88]:
```
df_train = df_train.drop(['Alone','Person','Cabin'],1)
```

```
C:\Users\yashm\anaconda3\lib\site-packages\ipykernel_launcher.py:1: FutureWarning: In a future version of pandas all ar
guments of DataFrame.drop except for the argument 'labels' will be keyword-only
  """Entry point for launching an IPython kernel.
```

In [89]:
```
df_train.isna().sum()
```

Out[89]:
```
Survived      0
Pclass        0
Sex           0
Age         177
SibSp         0
Parch         0
Fare          0
Embarked      2
dtype: int64
```

In [90]:
```
df_train['Embarked'].value_counts()
```

Out[90]:
```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

In [92]:
```
df_train['Embarked'].fillna('S',inplace = True)
```

In [95]:
```
df_train['Age'].fillna(df_train['Age'].mean(),inplace = True)
```

In [96]:
```
df_train
```

Out[96]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.000000	1	0	7.2500	S
1	1	1	female	38.000000	1	0	71.2833	C
2	1	3	female	26.000000	0	0	7.9250	S
3	1	1	female	35.000000	1	0	53.1000	S
4	0	3	male	35.000000	0	0	8.0500	S
...	...	...	...	...	...	...	...	...
886	0	2	male	27.000000	0	0	13.0000	S
887	1	1	female	19.000000	0	0	30.0000	S
888	0	3	female	29.699118	1	2	23.4500	S
889	1	1	male	26.000000	0	0	30.0000	C
890	0	3	male	32.000000	0	0	7.7500	Q

891 rows × 8 columns

In [98]:
```
df_train = pd.get_dummies(df_train,drop_first=True)
```

In [102]:
```
from sklearn.model_selection import train_test_split
```

In [106]:
```
X_train,X_test,y_train,y_test = train_test_split(df_train.drop('Survived',1),df_train['Survived']
                                                 ,stratify=df_train['Survived'])
```

```
C:\Users\yashm\anaconda3\lib\site-packages\ipykernel_launcher.py:1: FutureWarning: In a future version of pandas all ar
guments of DataFrame.drop except for the argument 'labels' will be keyword-only
  """Entry point for launching an IPython kernel.
```

In [100]:
```
from sklearn.preprocessing import StandardScaler
```

In [107]:
```
scaler = StandardScaler()
```

In [109]:
```
X_train[['Age','Fare']] = scaler.fit_transform(X_train[['Age','Fare']])
```

In [111]:
```
X_test[['Age','Fare']] = scaler.transform(X_test[['Age','Fare']])
```

In [112]:
```
X_train
```

Out[112]:

	Pclass	Age	SibSp	Parch	Fare	Sex_male	Embarked_Q	Embarked_S
778	3	0.010684	0	0	-0.469543	1	1	0
272	2	0.852969	0	1	-0.239754	0	0	1
253	3	0.033110	1	0	-0.306175	1	0	1
767	3	0.070376	0	0	-0.469299	0	1	0
779	1	1.002034	0	1	3.507934	0	0	1
...	...	...	...	...	...	...	...	...
881	3	0.256708	0	0	-0.466450	1	0	1
125	3	-1.308478	1	0	-0.401086	1	0	0
21	2	0.331240	0	0	-0.366736	1	0	1
878	3	0.010684	0	0	-0.466450	1	0	1
707	1	0.927501	0	0	-0.107155	1	0	1

668 rows × 8 columns

In [113]:
```
y_train
```

Out[113]:
```
778    0
272    1
253    0
767    0
779    1
      ..
881    0
125    1
21     1
878    0
707    1
Name: Survived, Length: 668, dtype: int64
```

In [119]:

```
##
from sklearn.metrics import classification_report
model  = LogisticRegression()
model.fit(X_train,y_train)
print("Results of The Logistic Regression Model  is : \n")
print(classification_report(model.predict(X_test),y_test))
print("-------------------************-------------------------")

model  = KNeighborsClassifier()
model.fit(X_train,y_train)
print("Results of The Knn  is : \n")
print(classification_report(model.predict(X_test),y_test))
print("-------------------************-------------------------")

model  = DecisionTreeClassifier()
model.fit(X_train,y_train)
print("Results of The Decision Tree  is : \n")
print(classification_report(model.predict(X_test),y_test))
print("-------------------************-------------------------")

model  = RandomForestClassifier()
model.fit(X_train,y_train)
print("Results of The Random Forest  is : \n")
print(classification_report(model.predict(X_test),y_test))
print("-------------------************-------------------------")

model  = SVC()
model.fit(X_train,y_train)
print("Results of The Support Vector Machine is : \n")
print(classification_report(model.predict(X_test),y_test))
print("-------------------************-------------------------")
```

```
Results of The Logistic Regression Model  is : 

              precision    recall  f1-score   support

           0       0.83      0.79      0.81       144
           1       0.65      0.71      0.68        79

    accuracy                           0.76       223
```

```
   macro avg       0.74      0.75      0.75       223
weighted avg       0.77      0.76      0.76       223

-------------------************------------------------
Results of The Knn  is :

              precision    recall  f1-score   support

           0       0.82      0.80      0.81       142
           1       0.66      0.70      0.68        81

    accuracy                           0.76       223
   macro avg       0.74      0.75      0.75       223
weighted avg       0.77      0.76      0.76       223

-------------------************------------------------
Results of The Decision Tree  is :

              precision    recall  f1-score   support

           0       0.77      0.82      0.79       130
           1       0.72      0.67      0.69        93

    accuracy                           0.75       223
   macro avg       0.75      0.74      0.74       223
weighted avg       0.75      0.75      0.75       223

-------------------************------------------------
Results of The Random Forest  is :

              precision    recall  f1-score   support

           0       0.82      0.82      0.82       136
           1       0.72      0.71      0.72        87

    accuracy                           0.78       223
   macro avg       0.77      0.77      0.77       223
weighted avg       0.78      0.78      0.78       223

-------------------************------------------------
Results of The Support Vector Machine is :
```

```
              precision    recall  f1-score   support

           0       0.89      0.80      0.84       153
           1       0.64      0.79      0.71        70

    accuracy                           0.79       223
   macro avg       0.77      0.79      0.77       223
weighted avg       0.81      0.79      0.80       223

-------------------************------------------------
```

In [ ]:

```
1
```