In [1]:

```
import warnings
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('fivethirtyeight')
%matplotlib inline
warnings.filterwarnings('ignore')
```

In [2]:

```
train = pd.read_csv('C:\\Users\\MUSKAN\\Downloads\\train.csv')
train
```

Out[2]:

	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

In [3]:

```
test = pd.read_csv('C:\\Users\\MUSKAN\\Downloads\\test.csv')
test
```

Out[3]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...
413	1305	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 3236	8.0500	NaN	S
414	1306	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 17758	108.9000	C105	C
415	1307	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O.Q. 3101262	7.2500	NaN	S
416	1308	3	Ware, Mr. Frederick	male	NaN	0	0	359309	8.0500	NaN	S
417	1309	3	Peter, Master. Michael J	male	NaN	1	1	2668	22.3583	NaN	C

418 rows × 11 columns

In [4]:

```
train.shape
```

Out[4]:

```
(891, 12)
```

In [5]:

```
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]:

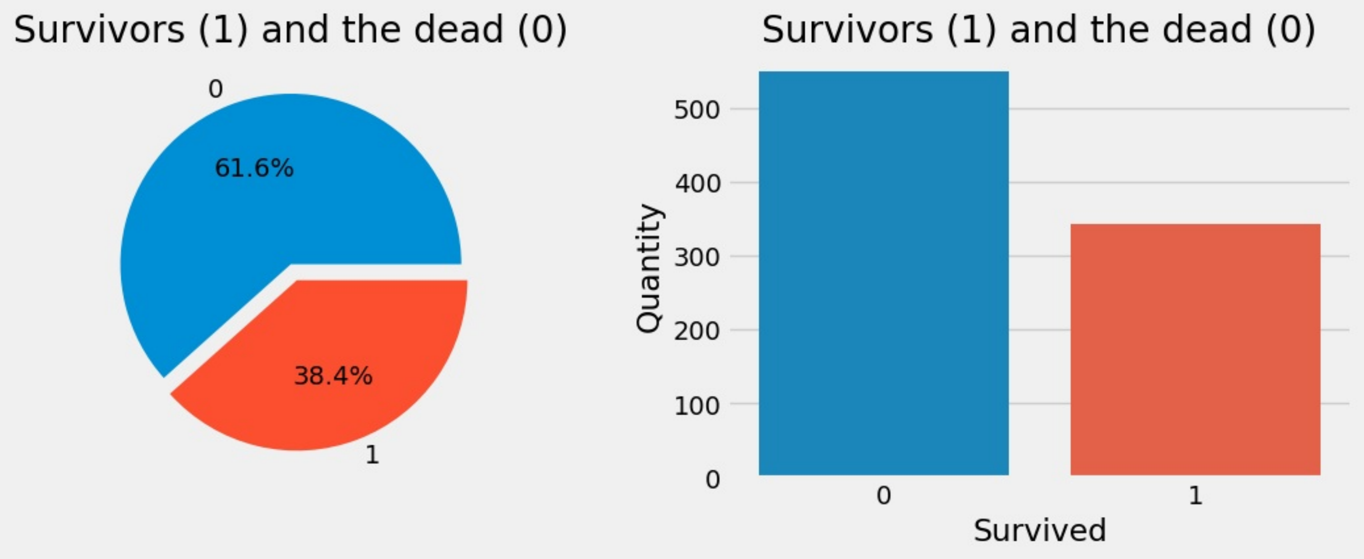
```
train.isnull().sum()
```

Out[6]:

```
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
```

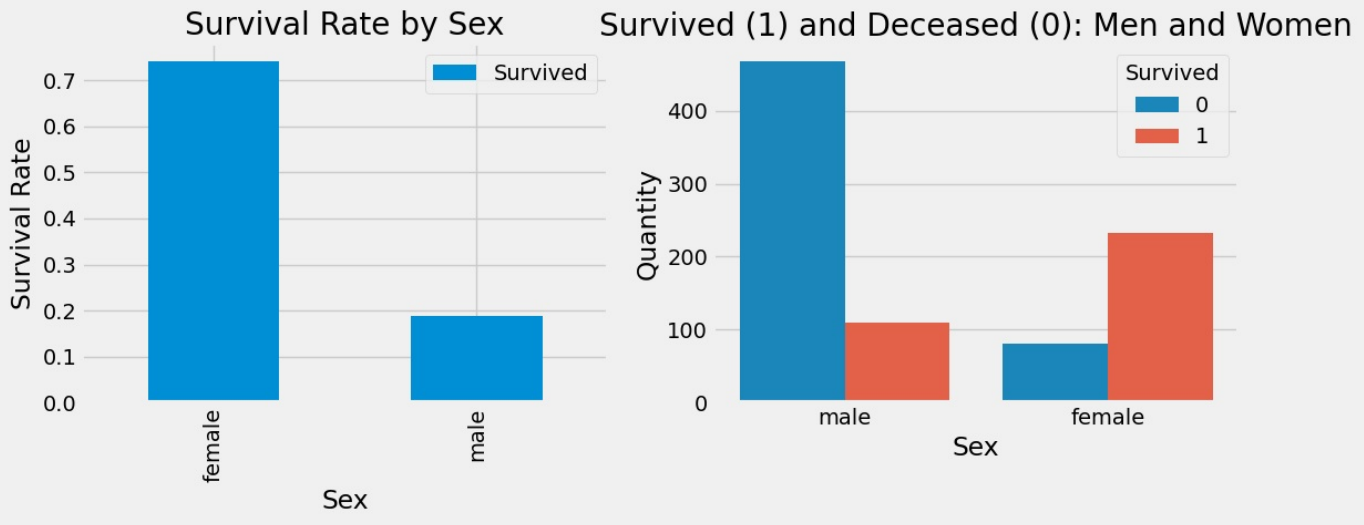
In [7]:

```
f, ax = plt.subplots(1, 2, figsize=(12, 4))
train['Survived'].value_counts().plot.pie(
    explode=[0, 0.1], autopct='%1.1f%%', ax=ax[0], shadow=False)
ax[0].set_title('Survivors (1) and the dead (0)')
ax[0].set_ylabel('')

sns.countplot(x='Survived', data=train, ax=ax[1])
ax[1].set_ylabel('Quantity')
ax[1].set_title('Survivors (1) and the dead (0)')

plt.show()
```



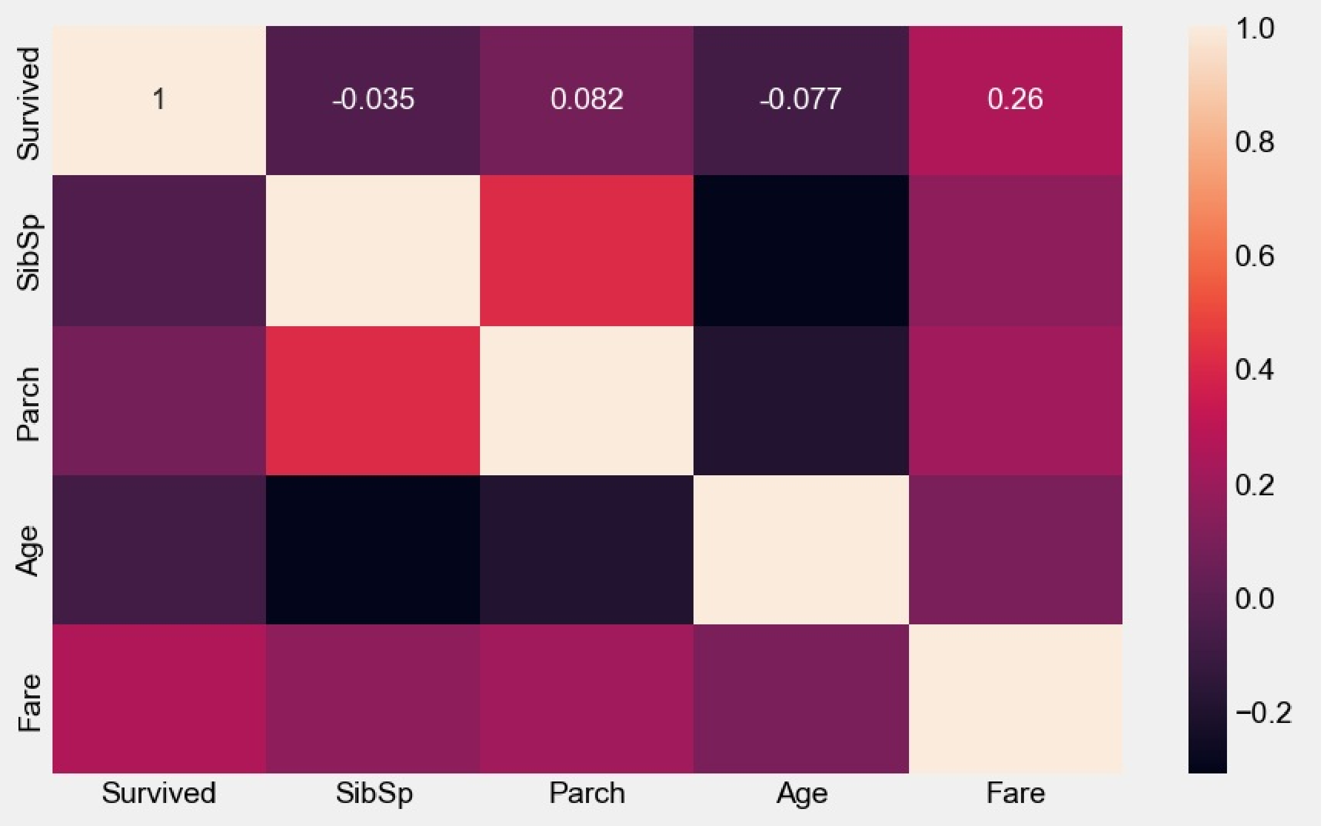
In [8]:

```
f, ax = plt.subplots(1, 2, figsize=(12, 4))
train[['Sex', 'Survived']].groupby(['Sex']).mean().plot.bar(ax=ax[0])
ax[0].set_title('Survival Rate by Sex')
ax[0].set_ylabel('Survival Rate')

sns.countplot(x='Sex', hue='Survived', data=train, ax=ax[1])
ax[1].set_ylabel('Quantity')
ax[1].set_title('Survived (1) and Deceased (0): Men and Women')

plt.show()
```



In [9]:

```
plt.figure(figsize=(10, 6))

# Create the heatmap of correlation
heatmap = sns.heatmap(train[["Survived", "SibSp", "Parch", "Age", "Fare"]].corr(), annot=True)
sns.set(rc={'figure.figsize': (8, 6)})
plt.show()
```



In [10]:

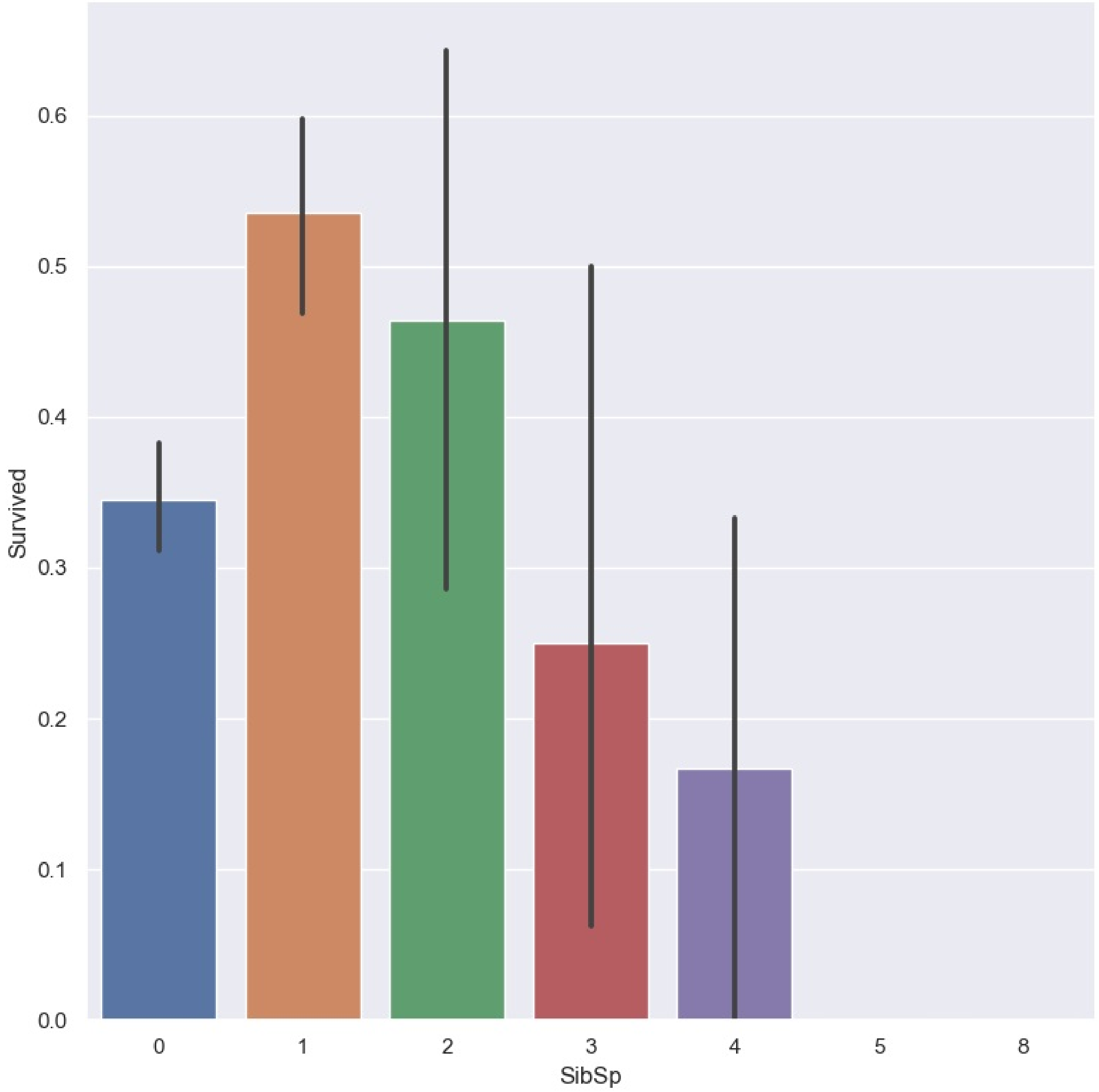
```
train['SibSp'].unique()
```

Out[10]:

```
array([1, 0, 3, 4, 2, 5, 8], dtype=int64)
```

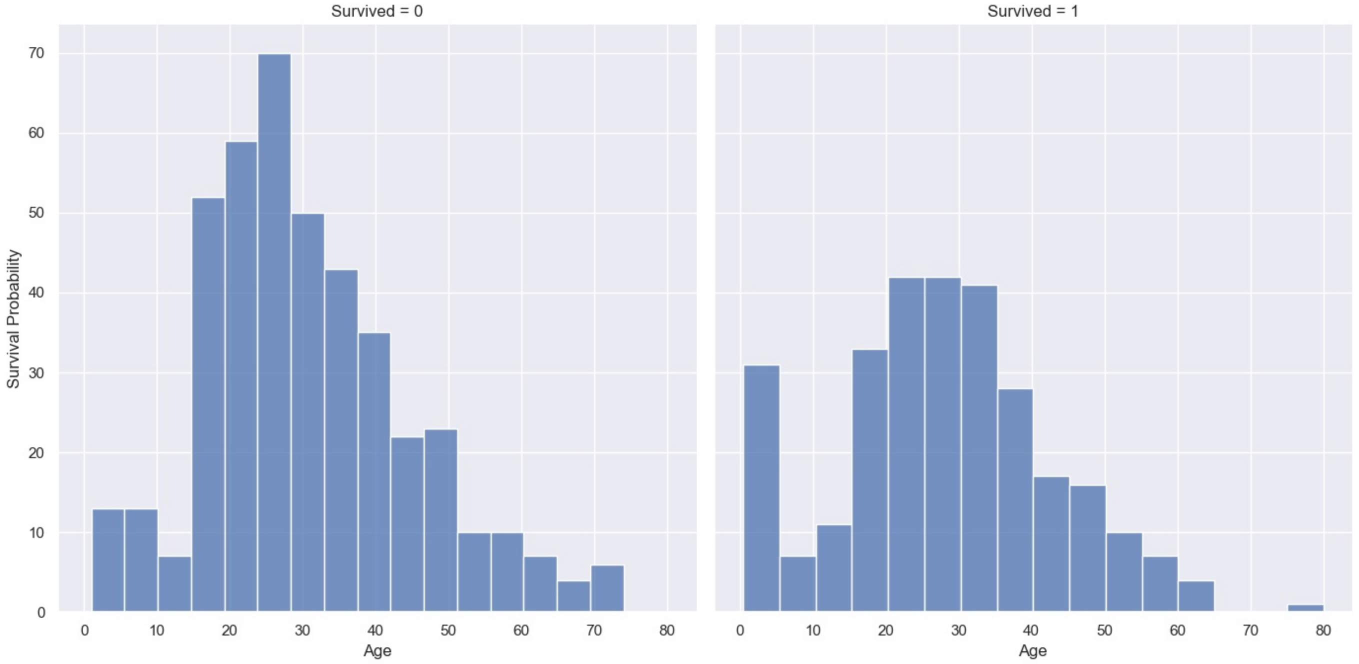
In [11]:

```
bargraph_sibsp = sns.catplot(x = "SibSp", y = "Survived", data = train, kind="bar", height = 8)
```



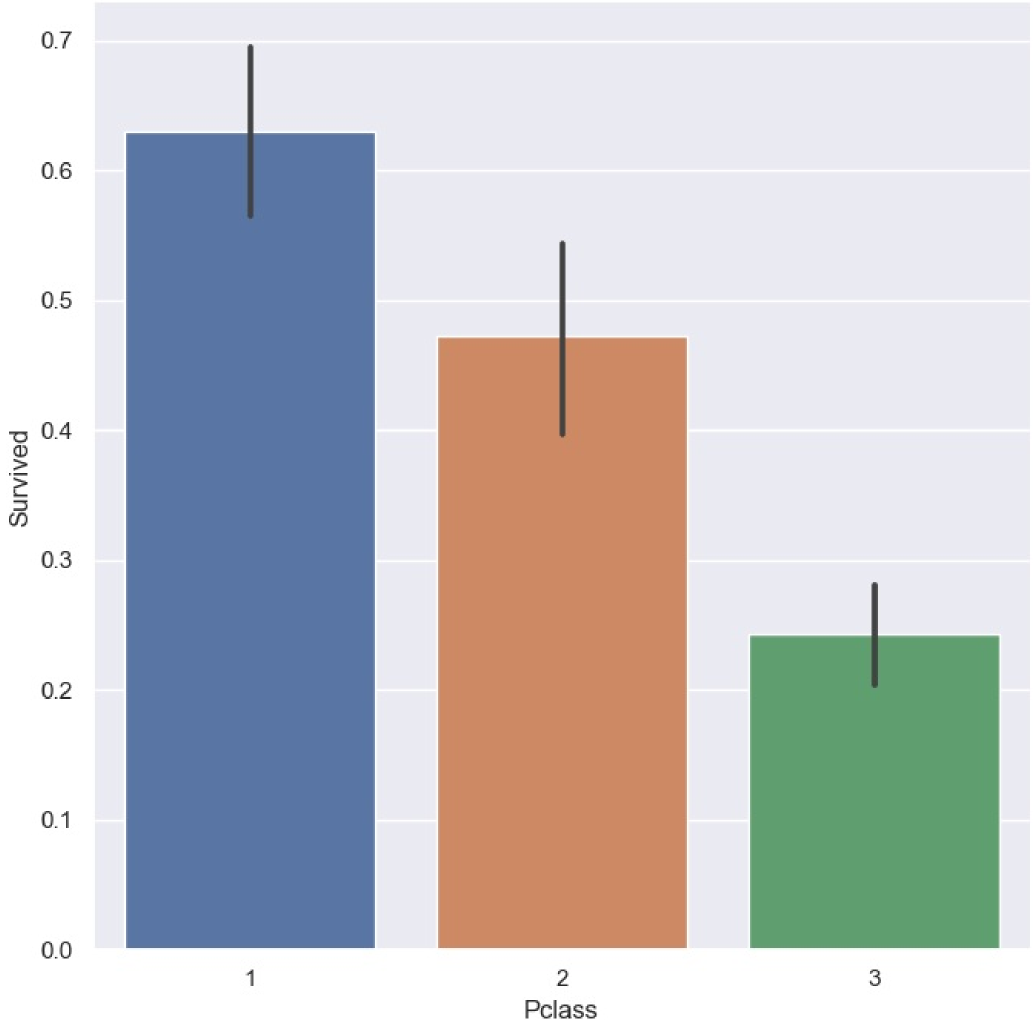
In [12]:

```
 age = sns.FacetGrid(train, col="Survived", height = 7)
age = age.map(sns.histplot, "Age")
age = age.set_ylabels("Survival Probability")
```



In [13]:

```
pclass = sns.catplot(x = "Pclass", y="Survived", data = train, kind="bar", height = 7)
```



In [14]:

```
train["CabinBool"] = (train["Cabin"].notnull().astype('int'))
test["CabinBool"] = (test["Cabin"].notnull().astype('int'))

train = train.drop(['Cabin'], axis=1)
test = test.drop(['Cabin'], axis=1)
```

In [15]:

```
train = train.drop(['Ticket'], axis=1)
test = test.drop(['Ticket'], axis=1)
```

In [16]:

```
train = train.fillna({"Embarked": "S"})
```

In [17]:

```
train["Age"] = train["Age"].fillna(-0.5)
test["Age"] = test["Age"].fillna(-0.5)
bins = [-1, 0, 5, 12, 18, 24, 35, 60, np.inf]
labels = ['Unknown', 'Baby', 'Child', 'Teenager',
                'Student', 'Young Adult', 'Adult', 'Senior']
train['AgeGroup'] = pd.cut(train["Age"], bins, labels=labels)
test['AgeGroup'] = pd.cut(test["Age"], bins, labels=labels)
```

In [18]:

```
# create a combined group of both datasets
combine = [train, test]

# extract a title for each Name in the
# train and test datasets
for dataset in combine:
        dataset['Title'] = dataset.Name.str.extract(' ([A-Za-z]+)\.', expand=False)

pd.crosstab(train['Title'], train['Sex'])

# replace various titles with more common names
for dataset in combine:
        dataset['Title'] = dataset['Title'].replace(['Lady', 'Capt', 'Col',
                                                                                                'Don', 'Dr', 'Ma
                                                                                                'Rev', 'Jonkhee
                                                                                                'Rare')


        dataset['Title'] = dataset['Title'].replace(
                ['Countess', 'Lady', 'Sir'], 'Royal')
        dataset['Title'] = dataset['Title'].replace('Mlle', 'Miss')
        dataset['Title'] = dataset['Title'].replace('Ms', 'Miss')
        dataset['Title'] = dataset['Title'].replace('Mme', 'Mrs')
```

```
train[['Title', 'Survived']].groupby(['Title'], as_index=False).mean()

# map each of the title groups to a numerical value
title_mapping = {"Mr": 1, "Miss": 2, "Mrs": 3,
                                "Master": 4, "Royal": 5, "Rare": 6}
for dataset in combine:
        dataset['Title'] = dataset['Title'].map(title_mapping)
        dataset['Title'] = dataset['Title'].fillna(0)
```

In [19]:

```
mr_age = train[train["Title"] == 1]["AgeGroup"].mode() # Young Adult
miss_age = train[train["Title"] == 2]["AgeGroup"].mode() # Student
mrs_age = train[train["Title"] == 3]["AgeGroup"].mode() # Adult
master_age = train[train["Title"] == 4]["AgeGroup"].mode() # Baby
royal_age = train[train["Title"] == 5]["AgeGroup"].mode() # Adult
rare_age = train[train["Title"] == 6]["AgeGroup"].mode() # Adult

age_title_mapping = {1: "Young Adult", 2: "Student",
                                        3: "Adult", 4: "Baby", 5: "Adult", 6: "Adult"}

for x in range(len(train["AgeGroup"])):
        if train["AgeGroup"][x] == "Unknown":
                train["AgeGroup"][x] = age_title_mapping[train["Title"][x]]

for x in range(len(test["AgeGroup"])):
        if test["AgeGroup"][x] == "Unknown":
                test["AgeGroup"][x] = age_title_mapping[test["Title"][x]]
```

In [20]:

```
# map each Age value to a numerical value
age_mapping = {'Baby': 1, 'Child': 2, 'Teenager': 3,
                        'Student': 4, 'Young Adult': 5, 'Adult': 6,
                        'Senior': 7}
train['AgeGroup'] = train['AgeGroup'].map(age_mapping)
test['AgeGroup'] = test['AgeGroup'].map(age_mapping)

train.head()

# dropping the Age feature for now, might change
train = train.drop(['Age'], axis=1)
test = test.drop(['Age'], axis=1)
```

In [21]:

```
train = train.drop(['Name'], axis=1)
test = test.drop(['Name'], axis=1)
```

In [22]:

```
sex_mapping = {"male": 0, "female": 1}
train['Sex'] = train['Sex'].map(sex_mapping)
test['Sex'] = test['Sex'].map(sex_mapping)

embarked_mapping = {"S": 1, "C": 2, "Q": 3}
train['Embarked'] = train['Embarked'].map(embarked_mapping)
test['Embarked'] = test['Embarked'].map(embarked_mapping)
test.head()
```

Out[22]:

	PassengerId	Pclass	Sex	SibSp	Parch	Fare	Embarked	CabinBool	AgeGroup	Title
0	892	3	0	0	0	7.8292	3	0	5.0	1
1	893	3	1	1	0	7.0000	1	0	6.0	3
2	894	2	0	0	0	9.6875	3	0	7.0	1
3	895	3	0	0	0	8.6625	1	0	5.0	1
4	896	3	1	1	1	12.2875	1	0	4.0	3

In [23]:

```
for x in range(len(test["Fare"])):
        if pd.isnull(test["Fare"][x]):
                pclass = test["Pclass"][x] # Pclass = 3
                test["Fare"][x] = round(
                        train[train["Pclass"] == pclass]["Fare"].mean(), 4)

# map Fare values into groups of
# numerical values
train['Fare'] = pd.qcut(train['Fare'], 4,
                                                        labels=[1, 2, 3, 4])
test['Fare'] = pd.qcut(test['Fare'], 4,
                                                labels=[1, 2, 3, 4])

# drop Fare values
train = train.drop(['Fare'], axis=1)
test = test.drop(['Fare'], axis=1)
test
```

Out[23]:

	PassengerId	Pclass	Sex	SibSp	Parch	Embarked	CabinBool	AgeGroup	Title
0	892	3	0	0	0	3	0	5.0	1
1	893	3	1	1	0	1	0	6.0	3
2	894	2	0	0	0	3	0	7.0	1
3	895	3	0	0	0	1	0	5.0	1
4	896	3	1	1	1	1	0	4.0	3
...	...	...	...	...	...	...	...	...	...
413	1305	3	0	0	0	1	0	5.0	1
414	1306	1	1	0	0	2	1	6.0	6
415	1307	3	0	0	0	1	0	6.0	1
416	1308	3	0	0	0	1	0	5.0	1
417	1309	3	0	1	1	2	0	1.0	4

418 rows × 9 columns

In [24]:

```
from sklearn.model_selection import train_test_split

# Drop the Survived and PassengerId
# column from the trainset
predictors = train.drop(['Survived', 'PassengerId'], axis=1)
target = train["Survived"]
x_train, x_val, y_train, y_val = train_test_split(
        predictors, target, test_size=0.2, random_state=0)
target
```

Out[24]:

```
0      0
1      1
2      1
3      1
4      0
      ..
886    0
887    1
888    0
889    1
890    0
Name: Survived, Length: 891, dtype: int64
```

In [25]:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

randomforest = RandomForestClassifier()

# Fit the training data along with its output
randomforest.fit(x_train, y_train)
y_pred = randomforest.predict(x_val)

# Find the accuracy score of the model
acc_randomforest = round(accuracy_score(y_pred, y_val) * 100, 2)
print(acc_randomforest)
```

```
81.01
```

In [26]:

```
ids = test['PassengerId']
predictions = randomforest.predict(test.drop('PassengerId', axis=1))

# set the output as a dataframe and convert
# to csv file named resultfile.csv
output = pd.DataFrame({'PassengerId': ids, 'Survived': predictions})
output.to_csv('resultfile.csv', index=False)
output
```

Out[26]:

	PassengerId	Survived
0	892	0
1	893	1
2	894	0
3	895	0
4	896	0
...	...	...
413	1305	0
414	1306	1
415	1307	0
416	1308	0
417	1309	1

418 rows × 2 columns

In [ ]:

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js