Titanic: Machine Learning from Disaster

Data Fields

- **Survival** - Survival. 0 = No, 1 = Yes
- **Pclass** - Ticket class. 1 = 1st, 2 = 2nd, 3 = 3rd
- **Sex** - Sex.
- **Age** - Age in years.
- **SibSp** - # of siblings / spouses aboard the Titanic.
- **Parch** - # of parents / children aboard the Titanic.
- **Ticket** - Ticket number.
- **Fare** - Passenger fare.
- **Cabin** - Cabin number.
- **Embarked** - Port of Embarkation. C = Cherbourg, Q = Queenstown, S = Southampton

01. 데이터 불러오기

In [2]:
```
import matplotlib
import matplotlib.pylab as pylab
import matplotlib.pyplot as plt
import matplotlib as mpl
import seaborn as sns

import pandas as pd
import numpy as np

import xgboost as xgb
import sklearn
import warnings

from sklearn.metrics import make_scorer, accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
import scipy
import numpy
import json
import sys
import csv
import os
```

In [3]:
```
# import train and test to play with it
df_train = pd.read_csv('data/train.csv')
df_test = pd.read_csv('data/test.csv')
```

In [48]:
```
print( type(df_train), type(df_test) )
```

```
<class 'pandas.core.frame.DataFrame'> <class 'pandas.core.frame.DataFrame'>
```

1-2 버전 확인

In [50]:
```
print('matplotlib: {}'.format(matplotlib.__version__))
print('sklearn: {}'.format(sklearn.__version__))
print('scipy: {}'.format(scipy.__version__))
print('seaborn: {}'.format(sns.__version__))
print('pandas: {}'.format(pd.__version__))
print('numpy: {}'.format(np.__version__))
print('Python: {}'.format(sys.version))
```

```
matplotlib: 2.1.2
sklearn: 0.19.1
scipy: 1.0.0
seaborn: 0.8.1
pandas: 0.22.0
numpy: 1.14.0
Python: 3.6.4 |Anaconda, Inc.| (default, Jan 16 2018, 10:22:32) [MSC v.1900 64 bit (AMD64)]
```

In [51]:
```
sns.set(style='white', context='notebook', palette='deep')
pylab.rcParams['figure.figsize'] = 12,8
warnings.filterwarnings('ignore')
mpl.style.use('ggplot')
sns.set_style('white')
%matplotlib inline
```

02. EDA

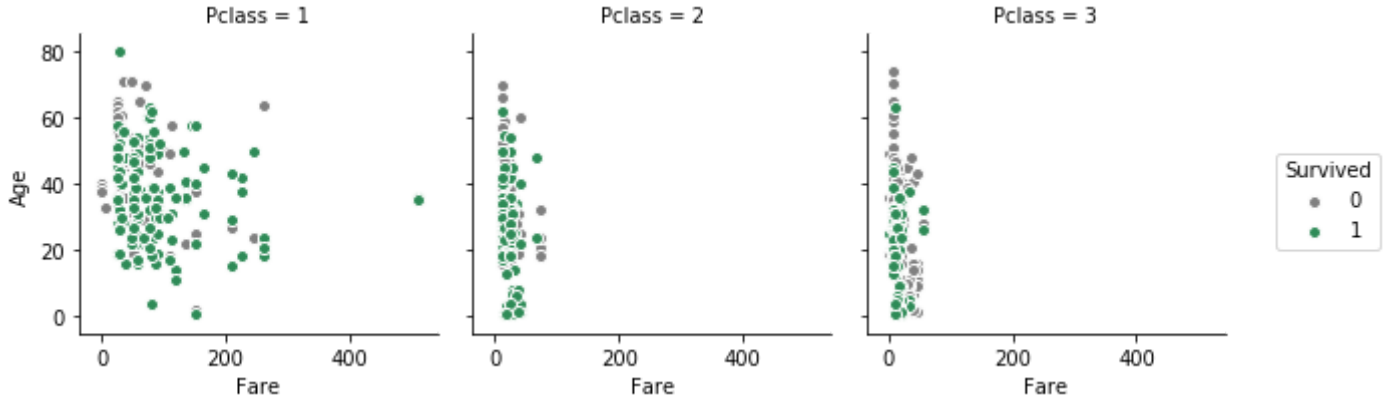
2-1 Scatter plot(산점도)

- 두 양적 변수간의 관계를 확인 목적을 갖습니다.

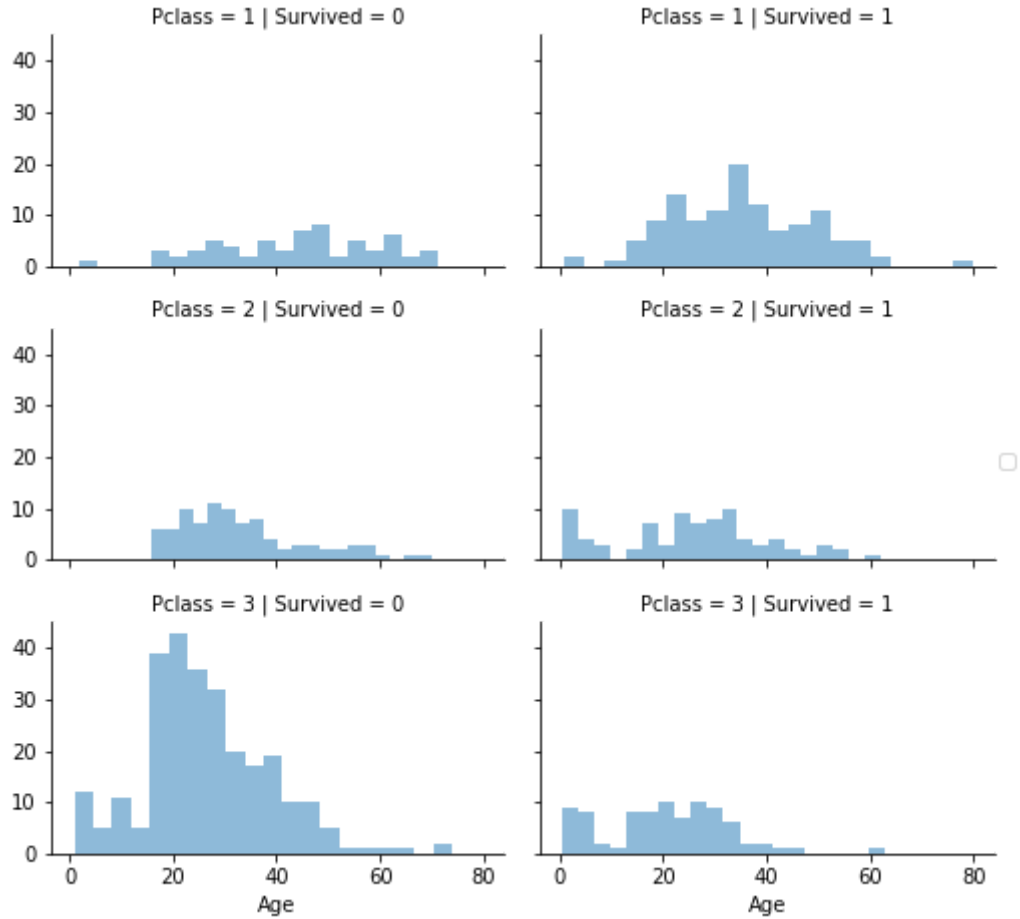
In [77]:
```
df_train.columns
```

Out[77]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [78]:
```
# Modify the graph above by assigning each species an individual color.
g = sns.FacetGrid(df_train, hue="Survived", col="Pclass", margin_titles=True,
                  palette={1:"seagreen", 0:"gray"})
g=g.map(plt.scatter, "Fare", "Age",edgecolor="w").add_legend();
```



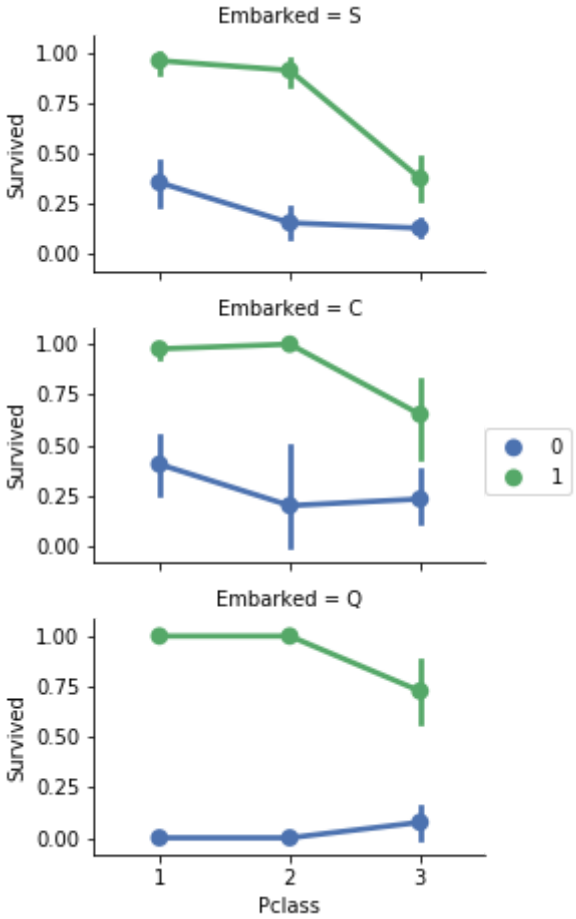
In [79]:
```
grid = sns.FacetGrid(df_train, col='Survived', row='Pclass', size=2.2, aspect=1.6)
grid.map(plt.hist, 'Age', alpha=.5, bins=20)
grid.add_legend();
```



In [80]:
```
# grid = sns.FacetGrid(train_df, col='Embarked')
grid = sns.FacetGrid(df_train, row='Embarked', size=2.2, aspect=1.6)
grid.map(sns.pointplot, 'Pclass', 'Survived', 'Sex', palette='deep')
grid.add_legend()
```

```
C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩seaborn₩axisgrid.py:703: UserWarning: Using the pointplot function without specifying
`order` is likely to produce an incorrect plot.
  warnings.warn(warning)
C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩seaborn₩axisgrid.py:708: UserWarning: Using the pointplot function without specifying
`hue_order` is likely to produce an incorrect plot.
  warnings.warn(warning)
```

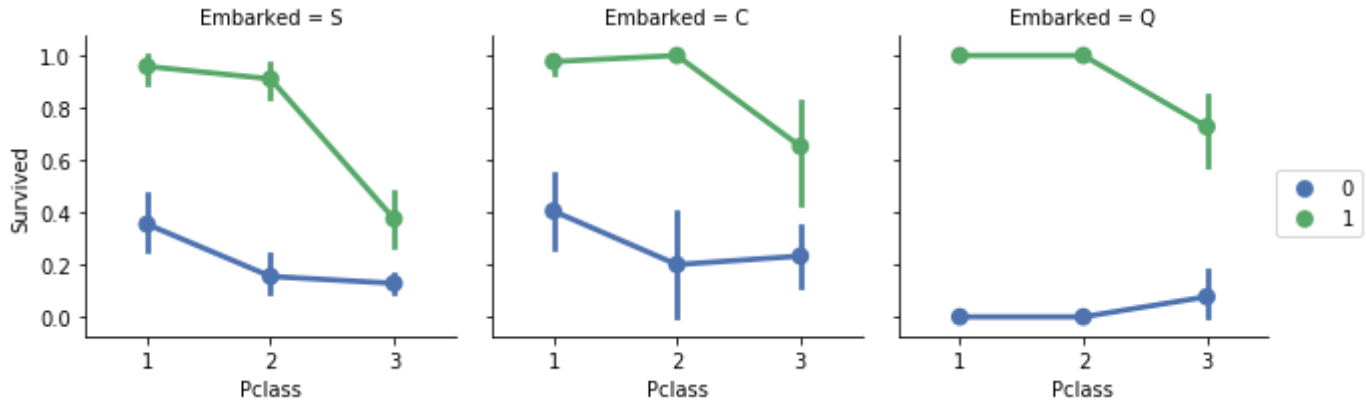
Out[80]:
```
<seaborn.axisgrid.FacetGrid at 0x1efd04a2710>
```


Embarked = S
Embarked = C
Embarked = Q
Survived
1.00
0.75
0.50
0.25
0.00
0
1
1
2
3
Pclass

In [81]:
```
e = sns.FacetGrid(df_train, col = 'Embarked')
e.map(sns.pointplot, 'Pclass', 'Survived', 'Sex', ci=95.0, palette = 'deep')
e.add_legend()
```

```
C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩seaborn₩axisgrid.py:703: UserWarning: Using the pointplot function without specifying
`order` is likely to produce an incorrect plot.
  warnings.warn(warning)
C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩seaborn₩axisgrid.py:708: UserWarning: Using the pointplot function without specifying
`hue_order` is likely to produce an incorrect plot.
  warnings.warn(warning)
```

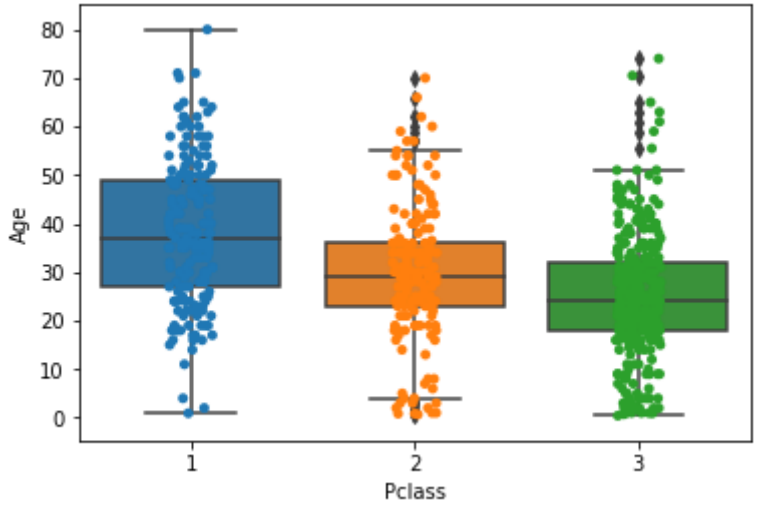
Out[81]: <seaborn.axisgrid.FacetGrid at 0x1efd2a0e860>



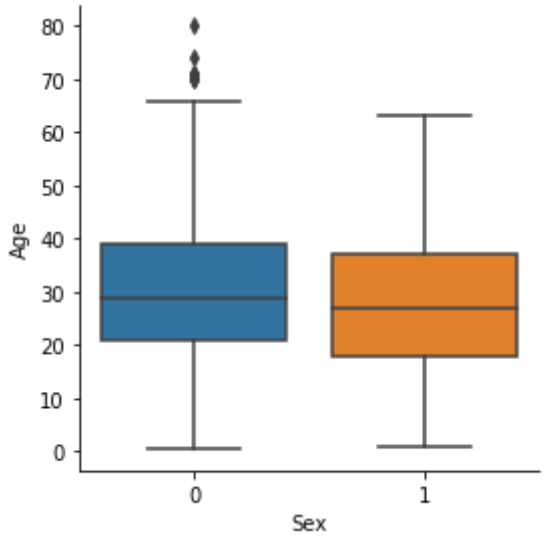
2-2 BoxPlot(상자 그림)

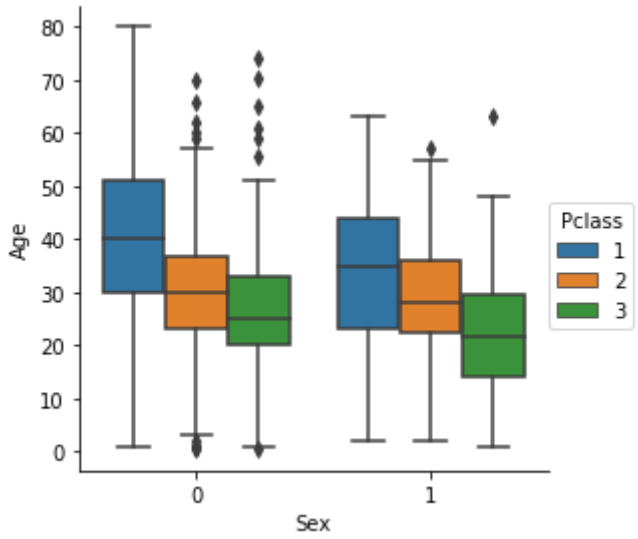
- 상자 그림은 사분위수를 통해 수치 데이터 그룹을 그래픽으로 묘사합니다.
- 이상치와 75%, 중앙값 25%의 값과 분포를 확인할 수 있습니다.

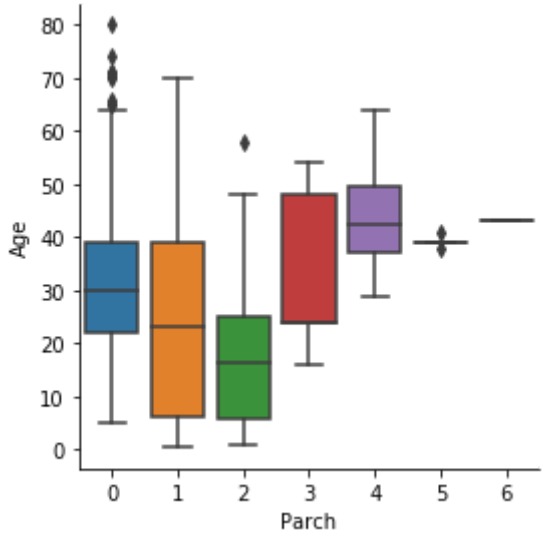
In [82]:
```
ax= sns.boxplot(x="Pclass", y="Age", data=df_train)
ax= sns.stripplot(x="Pclass", y="Age", data=df_train, jitter=True, edgecolor="gray")
plt.show()
```

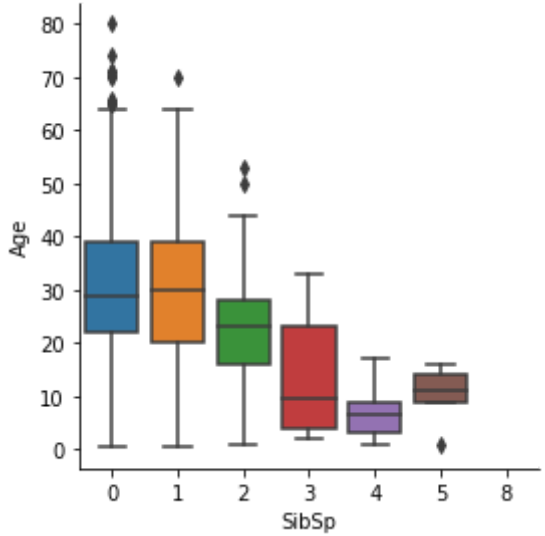


In [83]:
```
g = sns.factorplot(y="Age",x="Sex",data=df_train,kind="box")
g = sns.factorplot(y="Age",x="Sex",hue="Pclass", data=df_train,kind="box")
g = sns.factorplot(y="Age",x="Parch", data=df_train,kind="box")
g = sns.factorplot(y="Age",x="SibSp", data=df_train,kind="box")
```






Age
Parch
0
10
20
30
40
50
60
70
80
0
1
2
3
4
5
6


Age
SibSp
0
10
20
30
40
50
60
70
80
0
1
2
3
4
5
8

In [ ]:

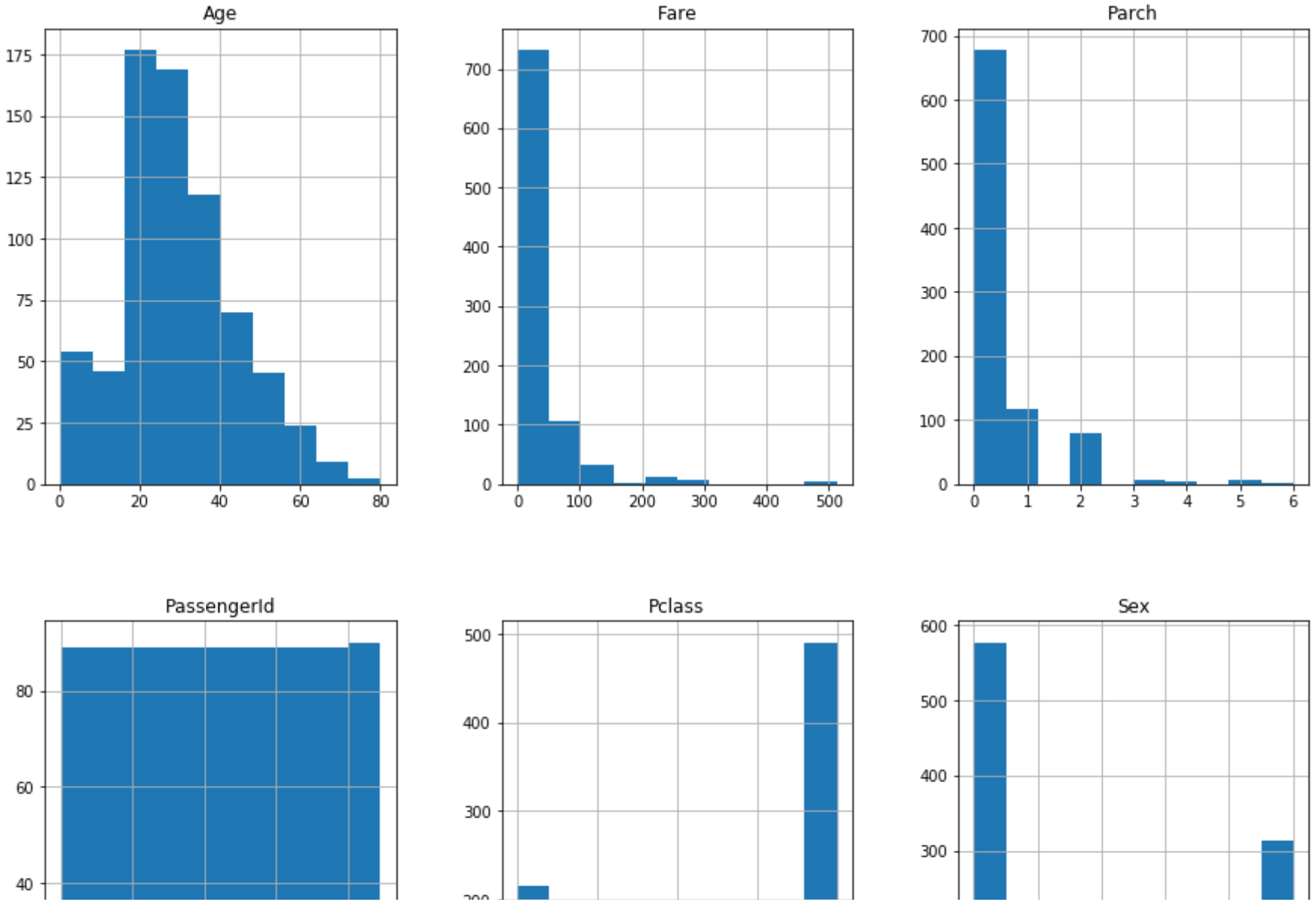
In [ ]:

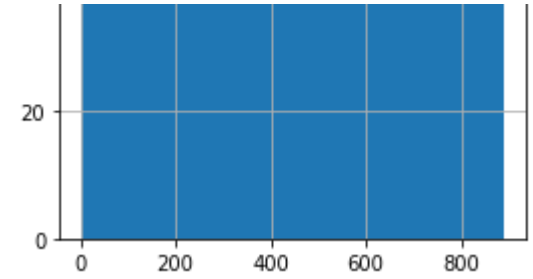
2-3 Histogram(히스토그램)

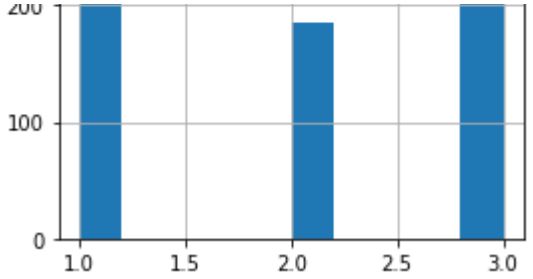
- 각각의 입력 변수에 대한 분포를 확인할 수 있습니다.

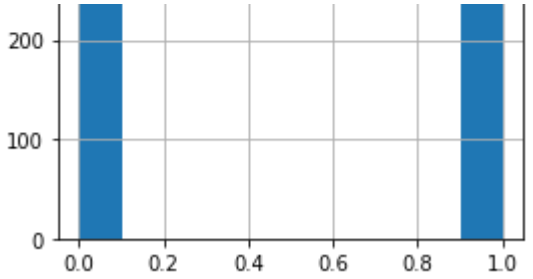
In [84]:
```
# histograms
df_train.hist(figsize=(15,20))
plt.figure()
```

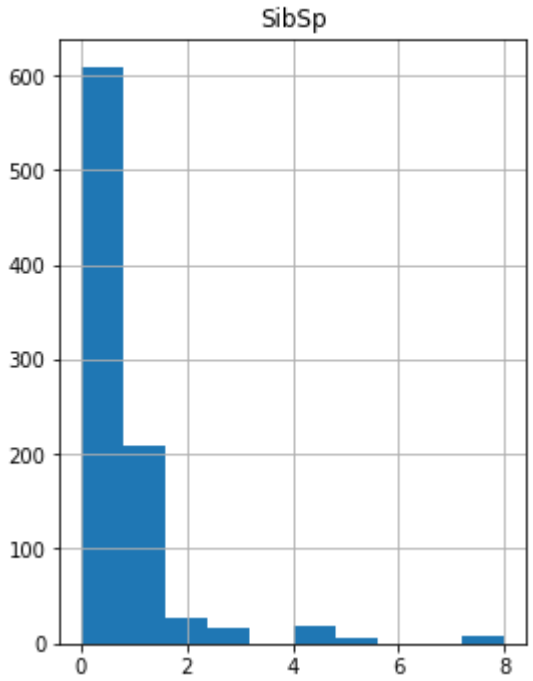
Out[84]: <matplotlib.figure.Figure at 0x1efd0a55160>

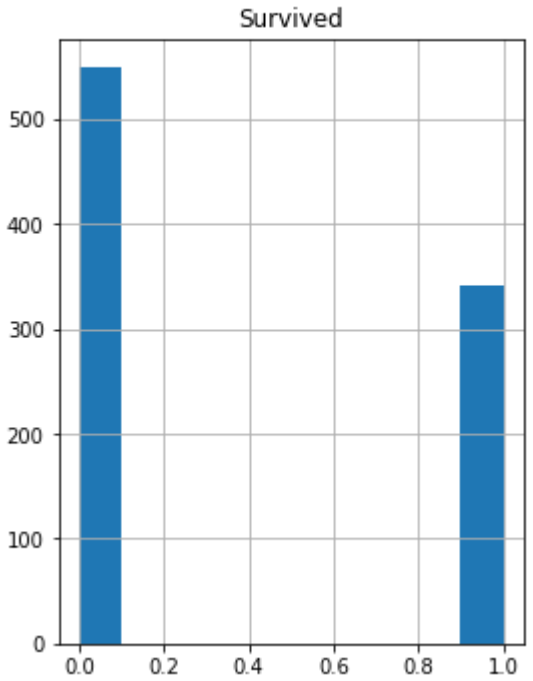








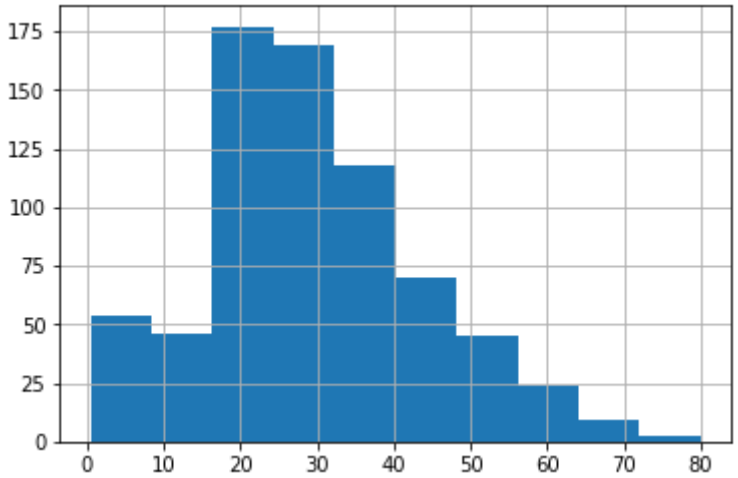




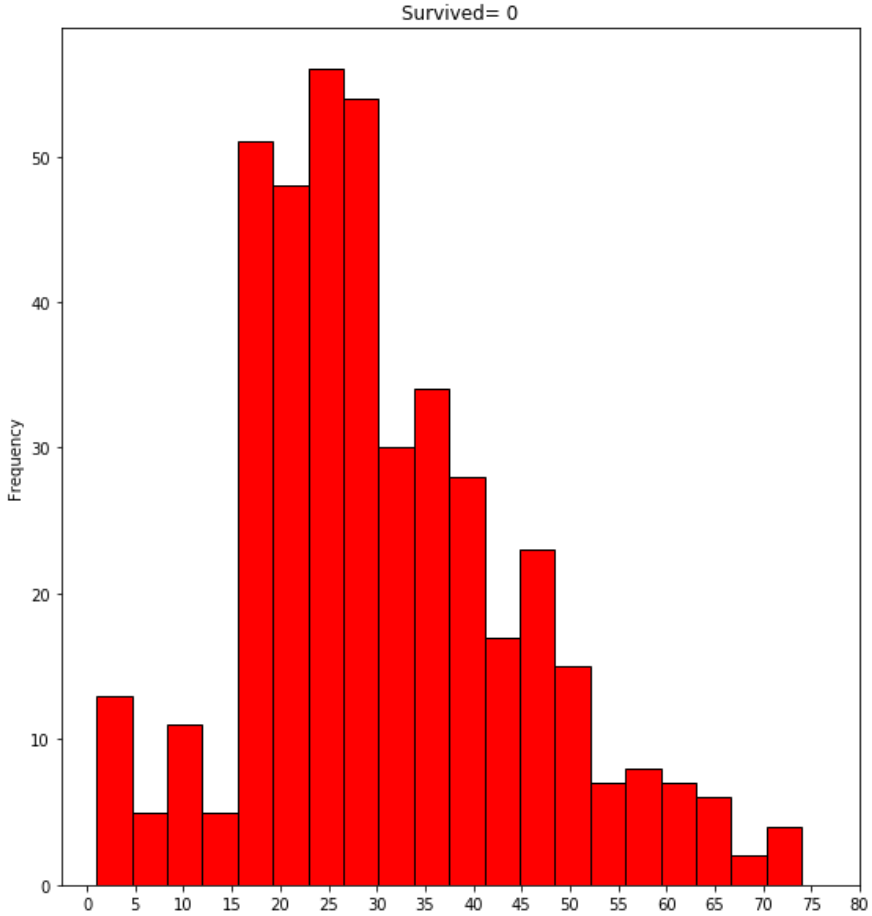

```
<matplotlib.figure.Figure at 0x1efd0a55160>
```

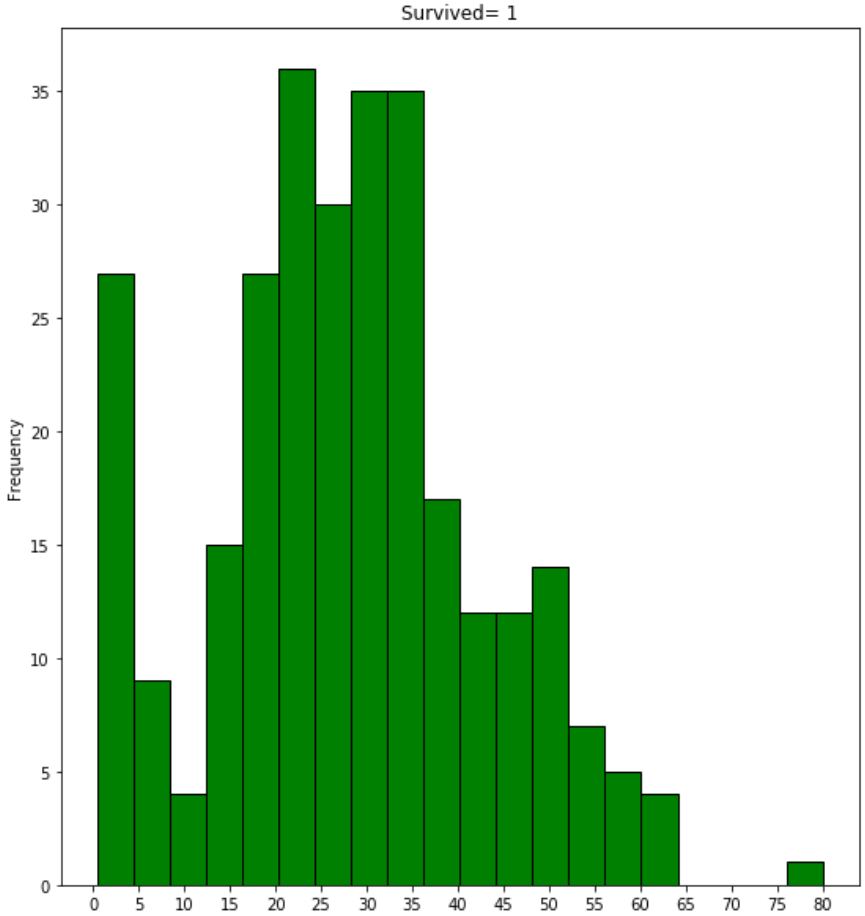
- Age가 가우시안 분포를 갖는 것 같습니다.

In [85]:
```
df_train["Age"].hist();
```



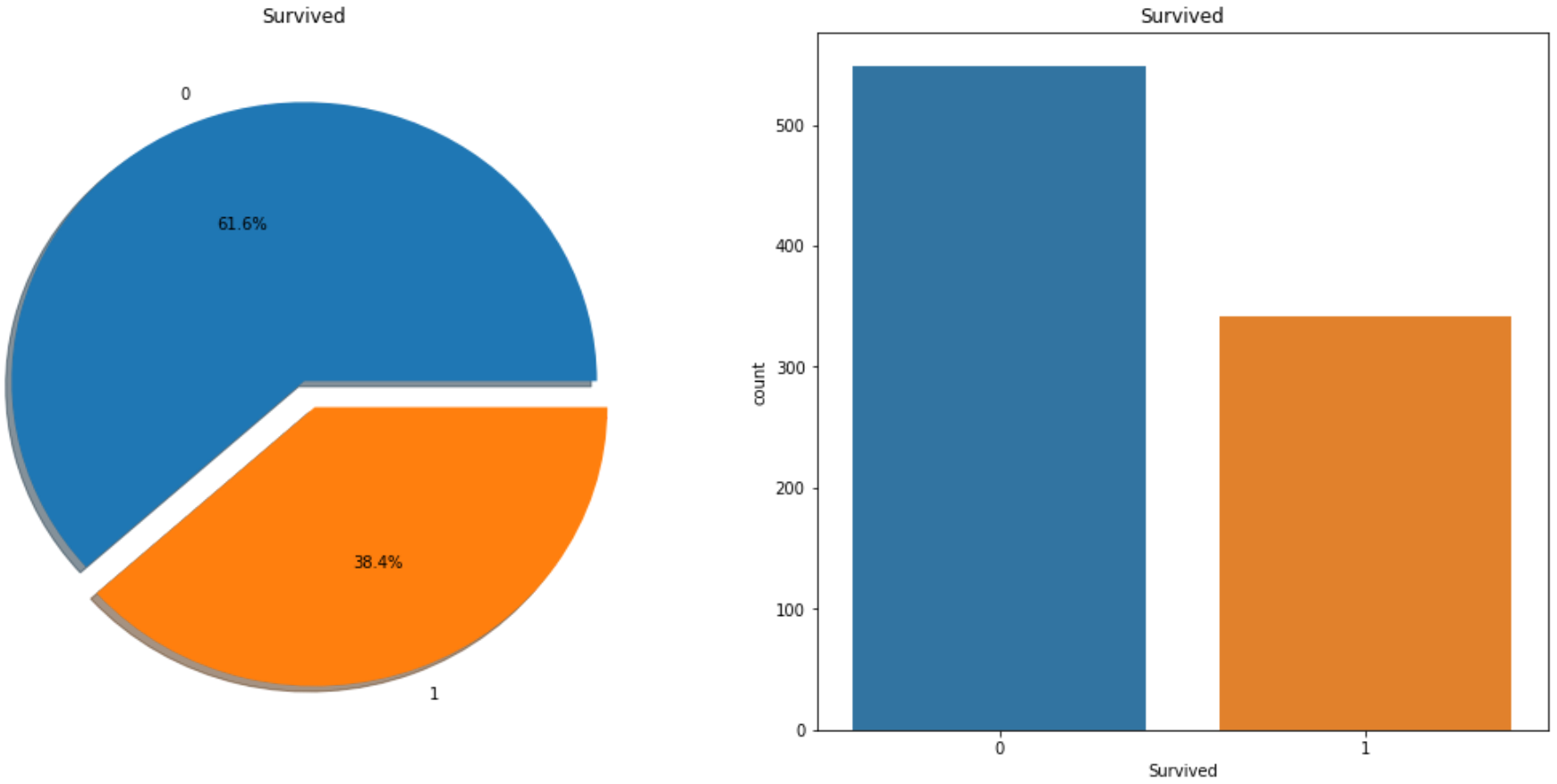
In [86]:
```
f,ax=plt.subplots(1,2,figsize=(20,10))
df_train[df_train['Survived']==0].Age.plot.hist(ax=ax[0],
                                                bins=20,edgecolor='black',color='red')

ax[0].set_title('Survived= 0')
x1=list(range(0,85,5))
ax[0].set_xticks(x1)     # 첫번째 그래프 x축 눈금
df_train[df_train['Survived']==1].Age.plot.hist(ax=ax[1],
                                                bins=20,edgecolor='black', color='green')

ax[1].set_title('Survived= 1')
x2=list(range(0,85,5))
ax[1].set_xticks(x2)     # 두번째 그래프 x축 눈금
plt.show()
```


Survived= 0
Frequency
0
10
20
30
40
50
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80

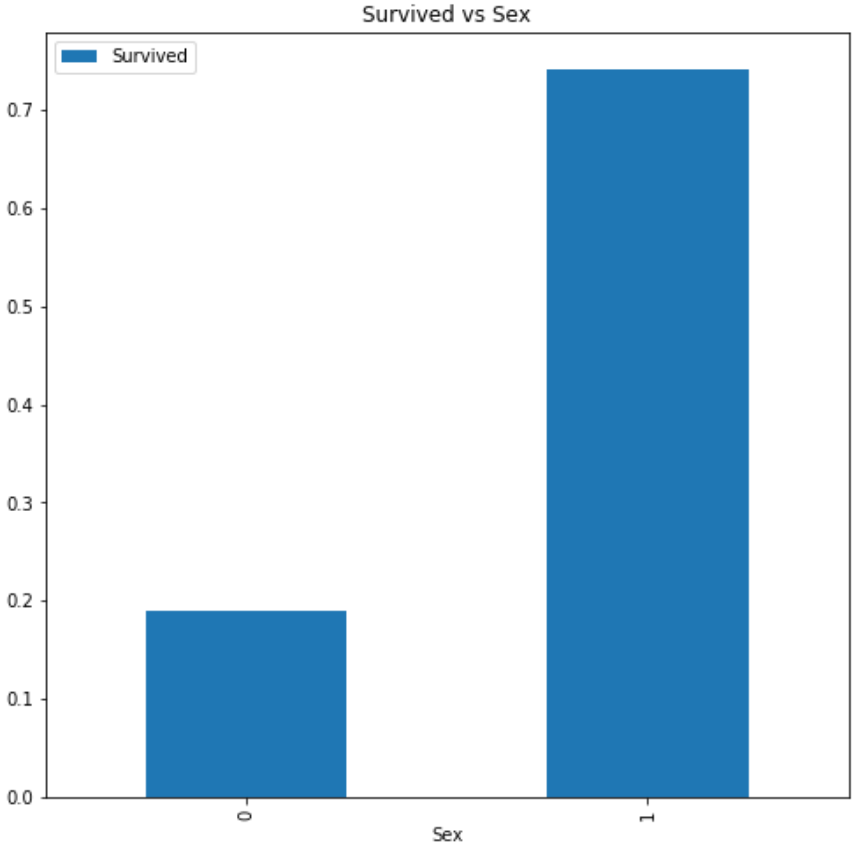

Survived= 1
Frequency
0
5
10
15
20
25
30
35
0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80

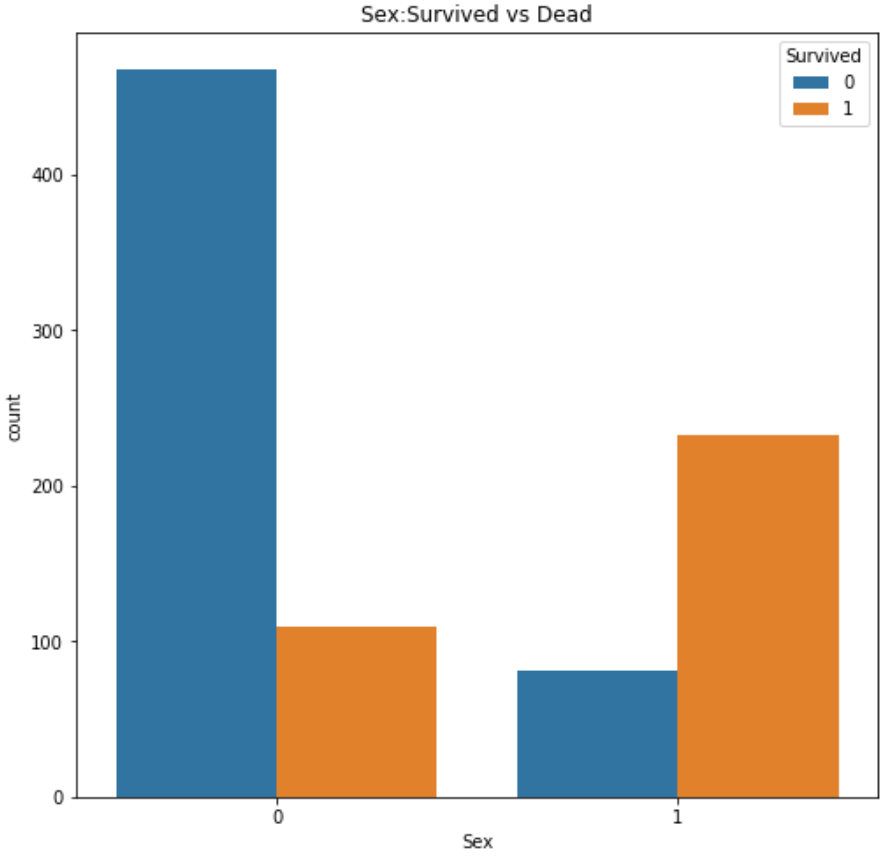
pie 그래프

In [87]:
```
f,ax=plt.subplots(1,2,figsize=(18,8))
df_train['Survived'].value_counts().plot.pie(explode=[0,0.1],
                                             autopct='%1.1f%%',ax=ax[0],shadow=True)
ax[0].set_title('Survived')
ax[0].set_ylabel('')

sns.countplot('Survived',data=df_train,ax=ax[1])
ax[1].set_title('Survived')
plt.show()
```



In [88]:
```
f,ax=plt.subplots(1,2,figsize=(18,8))

# 첫번째 그래프
df_train[['Sex','Survived']].groupby(['Sex']).mean().plot.bar(ax=ax[0])
ax[0].set_title('Survived vs Sex')

# 두번째 그래프
sns.countplot('Sex',hue='Survived',data=df_train,ax=ax[1])
ax[1].set_title('Sex:Survived vs Dead')
plt.show()
```



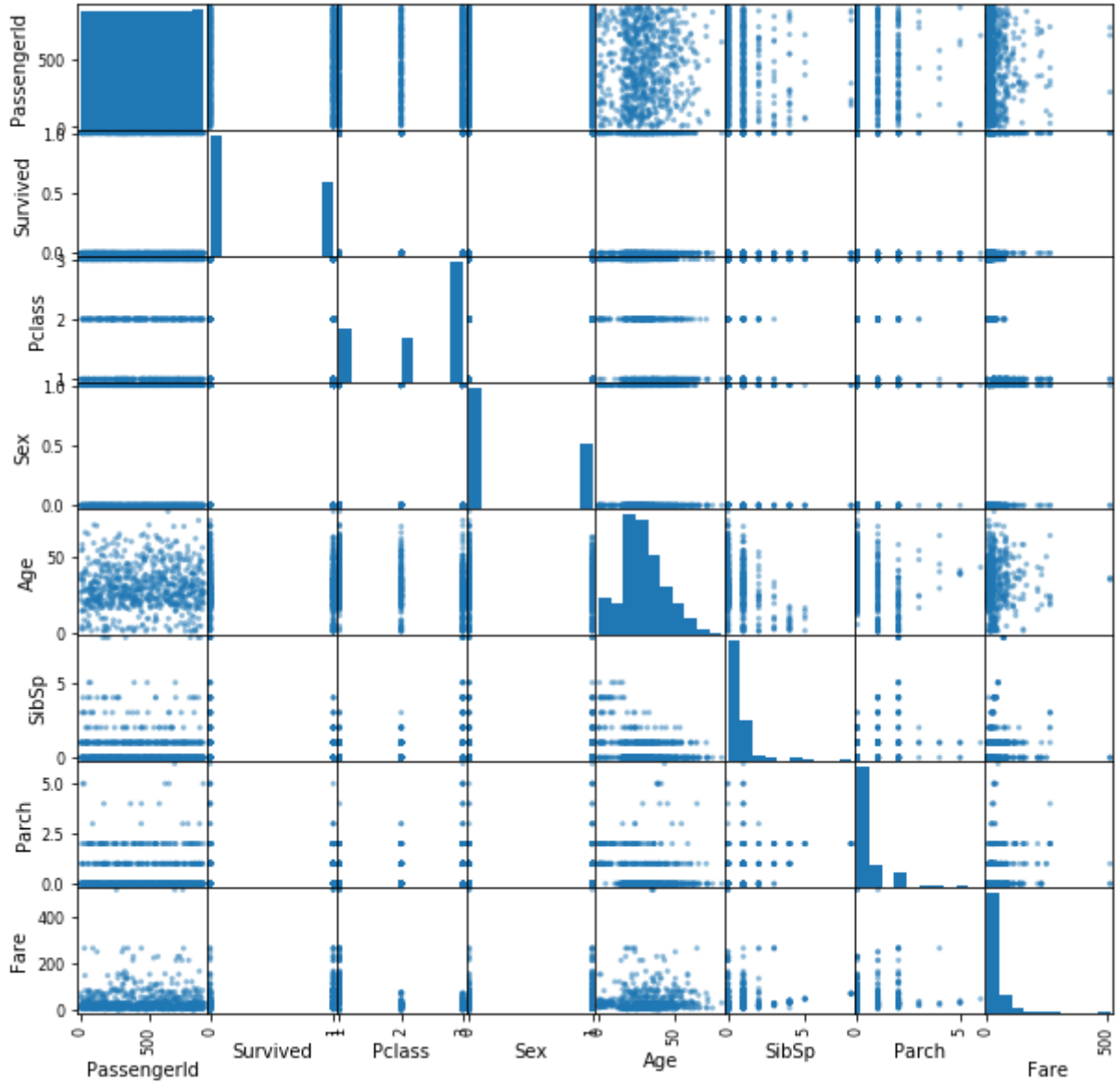


2-4 Multivariate Plots(다변량 플롯)

- 모든 속성 쌍의 산점도를 확인해 볼 수 있다.
- 입력 변수간의 구조화된 관계를 발견하는 데 도움이 될 수 있다.

In [89]:
```
# scatter plot matrix
pd.plotting.scatter_matrix(df_train,figsize=(10,10))
plt.figure()
```

Out[89]: <matplotlib.figure.Figure at 0x1efd052c5c0>

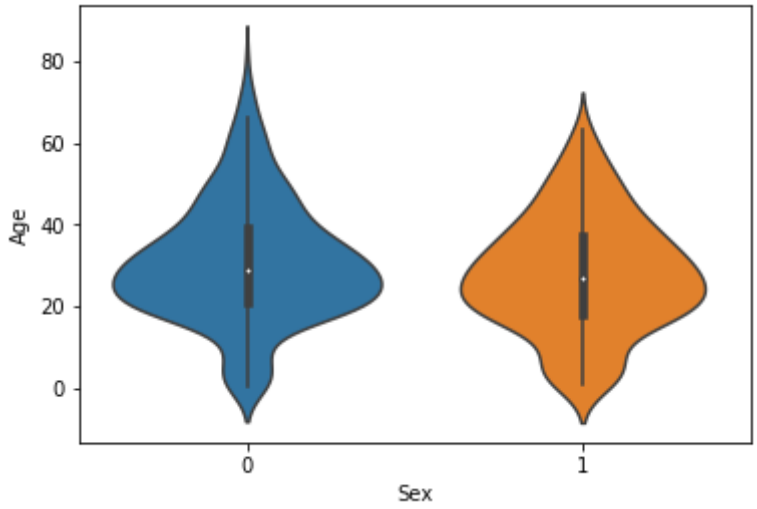


```
<matplotlib.figure.Figure at 0x1efd052c5c0>
```

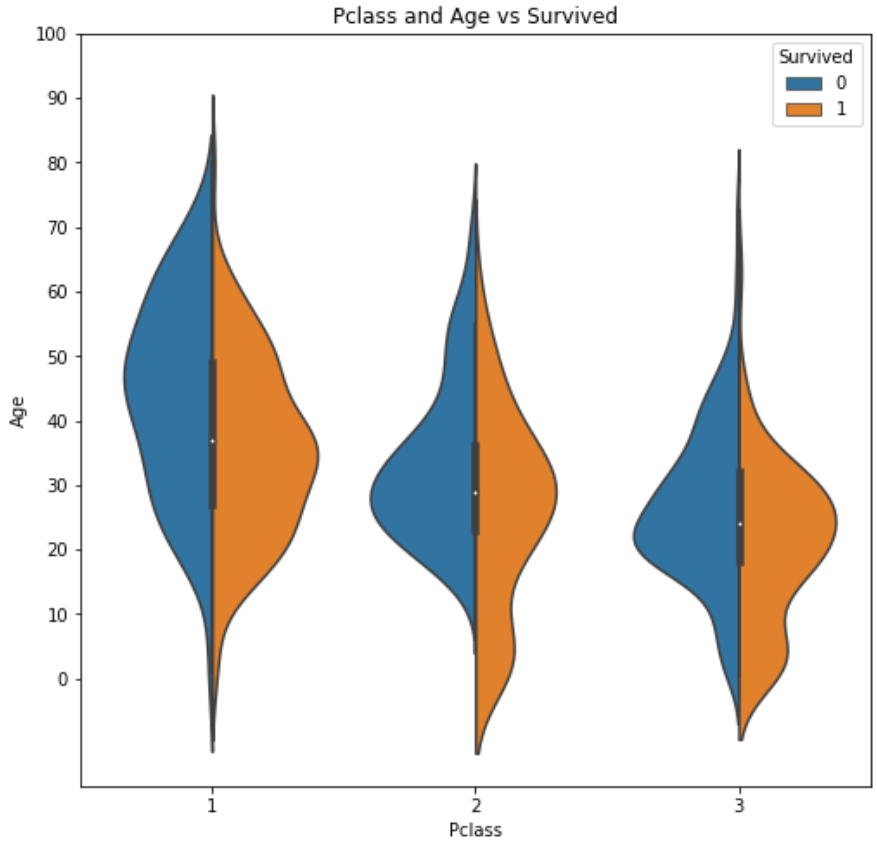
2-5 violinplots

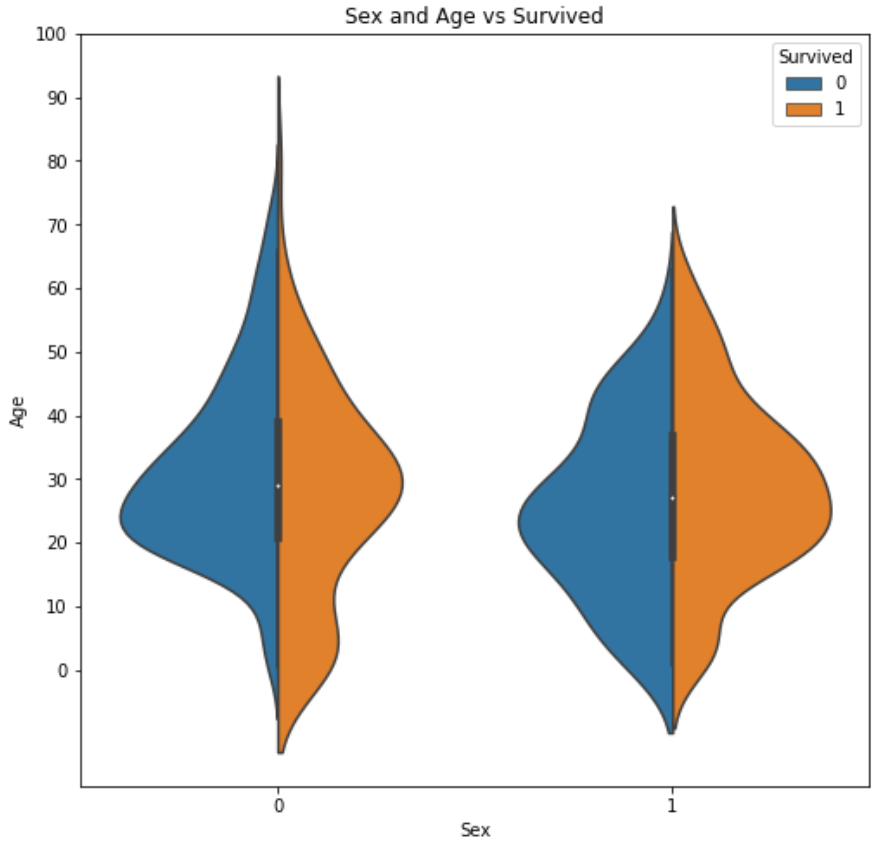
In [90]:
```
sns.violinplot(data=df_train,x="Sex", y="Age")
```

Out[90]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1efd0f3d1d0>
```



In [91]:
```
f,ax=plt.subplots(1,2,figsize=(18,8))

### 첫번째 그래프
sns.violinplot("Pclass","Age", hue="Survived", data=df_train,split=True,ax=ax[0])
ax[0].set_title('Pclass and Age vs Survived')
ax[0].set_yticks(range(0,110,10))

### 두번째 그래프
sns.violinplot("Sex","Age", hue="Survived", data=df_train,split=True,ax=ax[1])
ax[1].set_title('Sex and Age vs Survived')
ax[1].set_yticks(range(0,110,10))
plt.show()
```



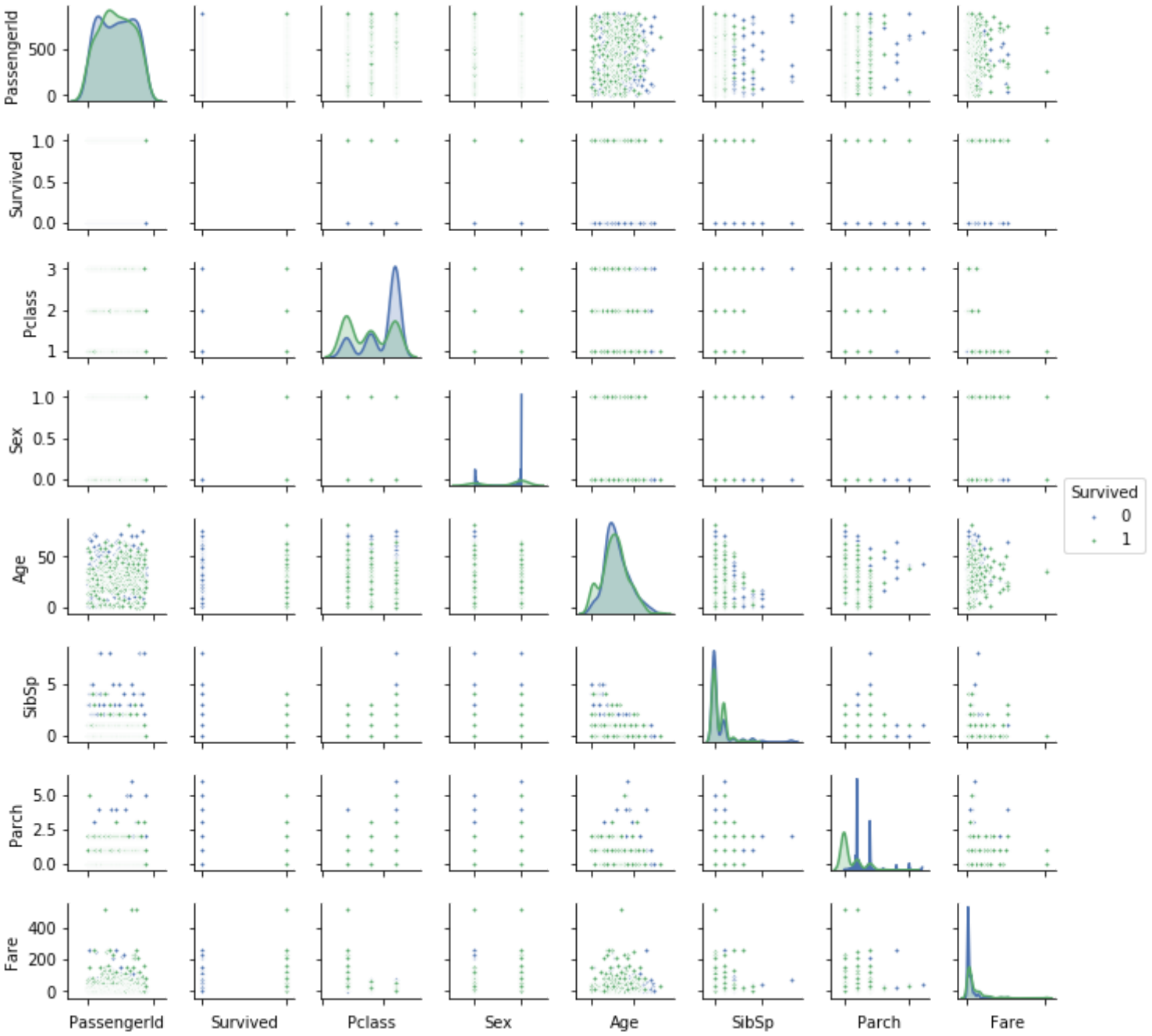


2-6 pairplot

In [92]:
```
# pair plots of entire dataset
pp = sns.pairplot(df_train, hue = 'Survived',
                  palette = 'deep',
                  size=1.2,
                  diag_kind = 'kde',
                  diag_kws=dict(shade=True),
                  plot_kws=dict(s=10) )

pp.set(xticklabels=[])
```

```
C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩statsmodels₩nonparametric₩kde.py:494: RuntimeWarning: invalid value encountered in tru
e_divide
  binned = fast_linbin(X,a,b,gridsize)/(delta*nobs)
C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩statsmodels₩nonparametric₩kdetools.py:34: RuntimeWarning: invalid value encountered in
double_scalars
  FAC1 = 2*(np.pi*bw/RANGE)**2
C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩numpy₩core₩_methods.py:26: RuntimeWarning: invalid value encountered in reduce
  return umr_maximum(a, axis, None, out, keepdims)
C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩statsmodels₩nonparametric₩kde.py:454: RuntimeWarning: invalid value encountered in gre
ater
  X = X[np.logical_and(X>clip[0], X<clip[1])] # won't work for two columns.
C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩statsmodels₩nonparametric₩kde.py:454: RuntimeWarning: invalid value encountered in les
s
  X = X[np.logical_and(X>clip[0], X<clip[1])] # won't work for two columns.
```

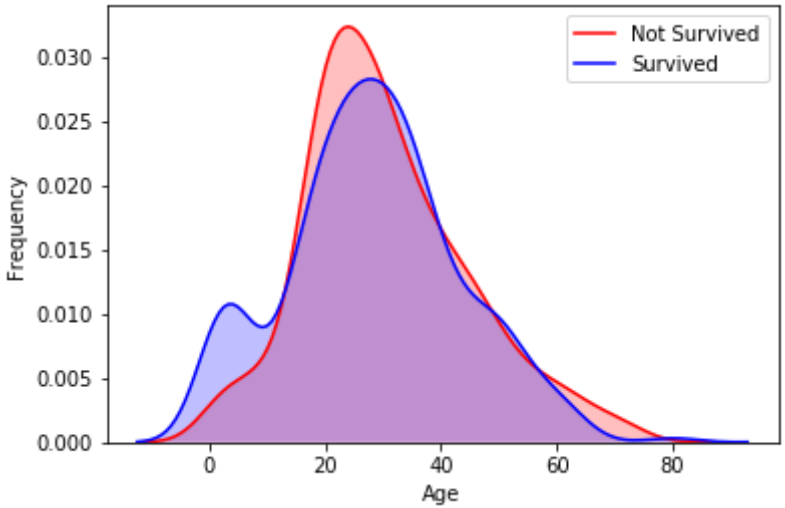
Out[92]: <seaborn.axisgrid.PairGrid at 0x1efd0eccc88>


PassengerId
Survived
Pclass
Sex
Age
SibSp
Parch
Fare
Survived
0
1

2-7 kdeplot

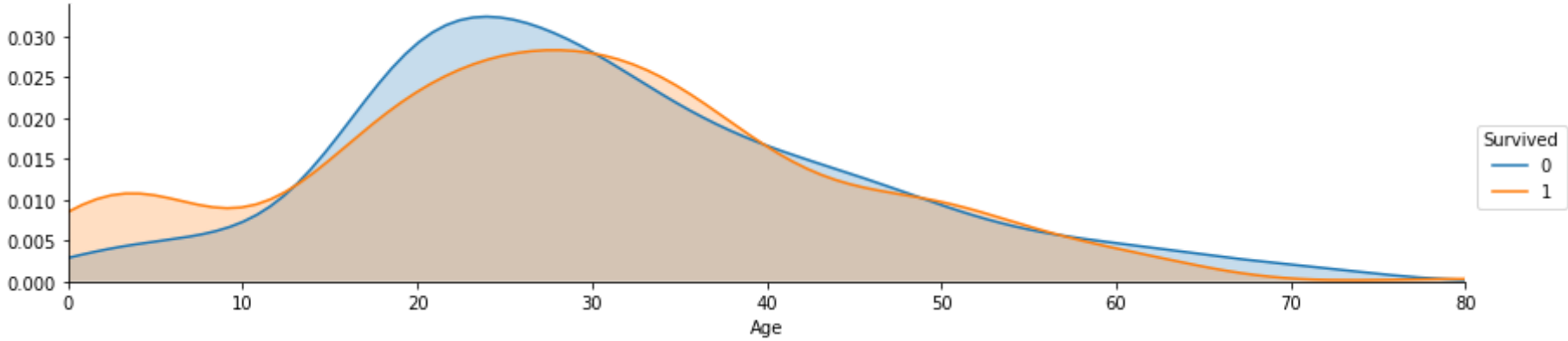
- pairplot의 막대그래프의 대각선에 표시된 막대 그래프를 kde로 대체 가능합니다.

In [93]:
```
# Explore Age distibution
g = sns.kdeplot(df_train["Age"][(df_train["Survived"] == 0) & (df_train["Age"].notnull())],
                color="Red", shade = True)
g = sns.kdeplot(df_train["Age"][(df_train["Survived"] == 1) & (df_train["Age"].notnull())],
                ax =g, color="Blue", shade= True)
g.set_xlabel("Age")
g.set_ylabel("Frequency")
g = g.legend(["Not Survived","Survived"])
```

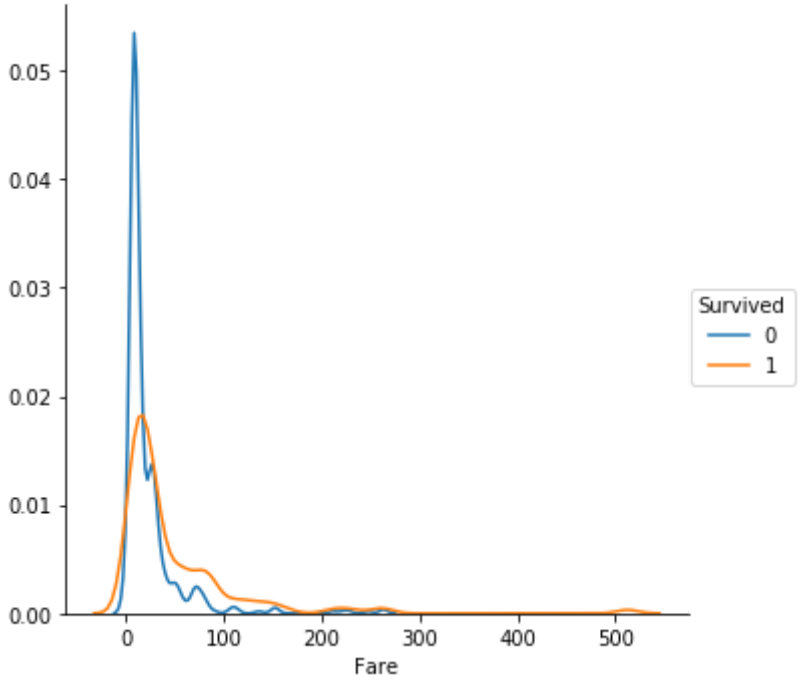


In [94]:
```
#plot distributions of age of passengers who survived or did not survive
a = sns.FacetGrid( df_train, hue = 'Survived', aspect=4 )
a.map(sns.kdeplot, 'Age', shade= True )
a.set(xlim=(0 , df_train['Age'].max()))
a.add_legend()
```

Out[94]: <seaborn.axisgrid.FacetGrid at 0x1efd09e07f0>



In [95]:
```
# seaborn's kdeplot, plots univariate or bivariate density estimates.
#Size can be changed by tweeking the value used
sns.FacetGrid(df_train, hue="Survived", size=5).map(sns.kdeplot, "Fare").add_legend()
plt.show()
```

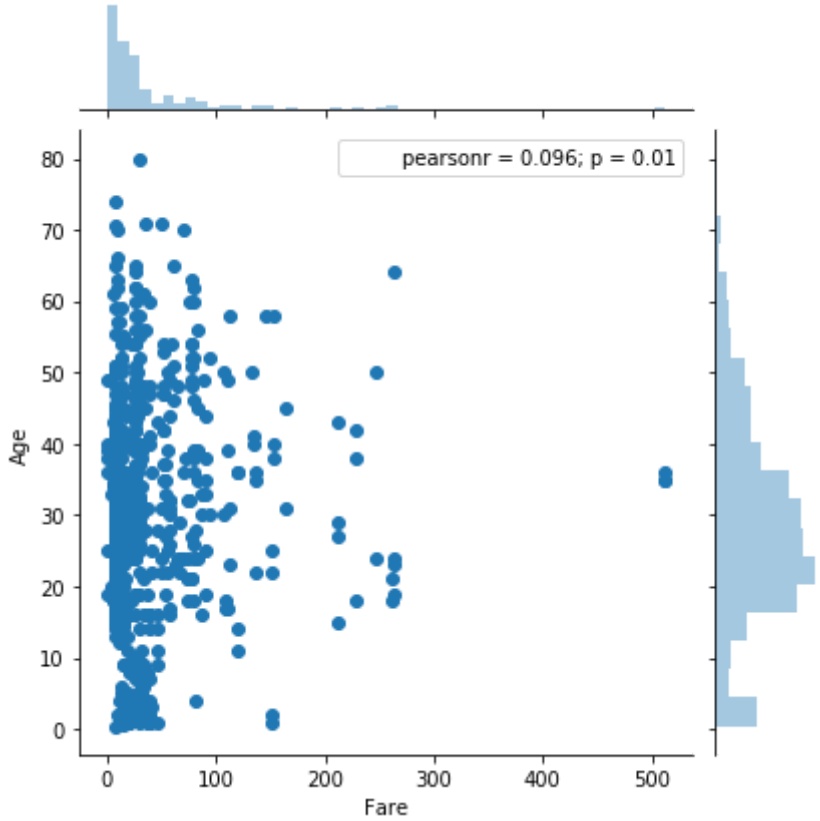


In [ ]:
```

```

2-8 jointplot

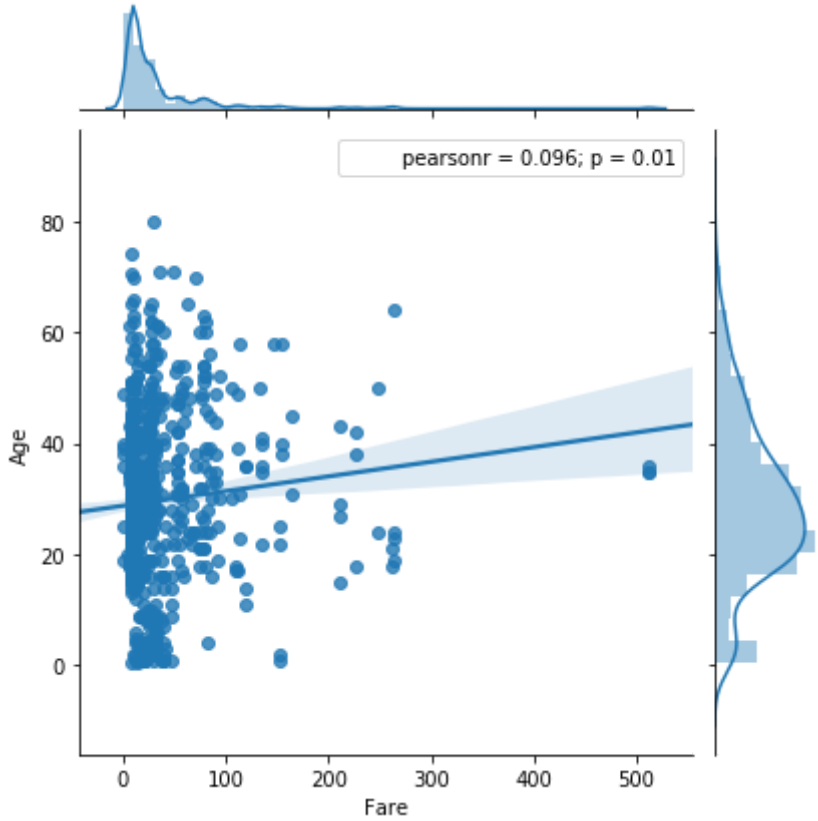
In [96]:
```
sns.jointplot(x='Fare',y='Age',data=df_train)
```

Out[96]: <seaborn.axisgrid.JointGrid at 0x1efd086b9b0>



In [97]:
```
sns.jointplot(x='Fare',y='Age' ,data=df_train, kind='reg')
```

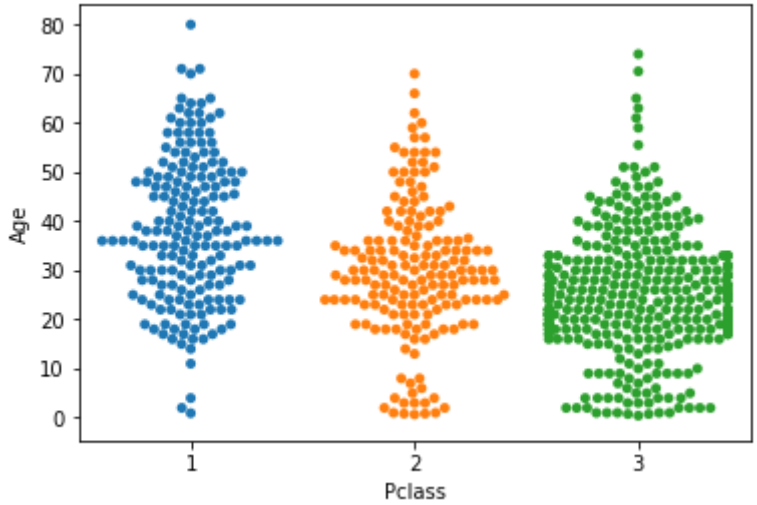
Out[97]: <seaborn.axisgrid.JointGrid at 0x1efd04a9320>



2-9 Swarm plot

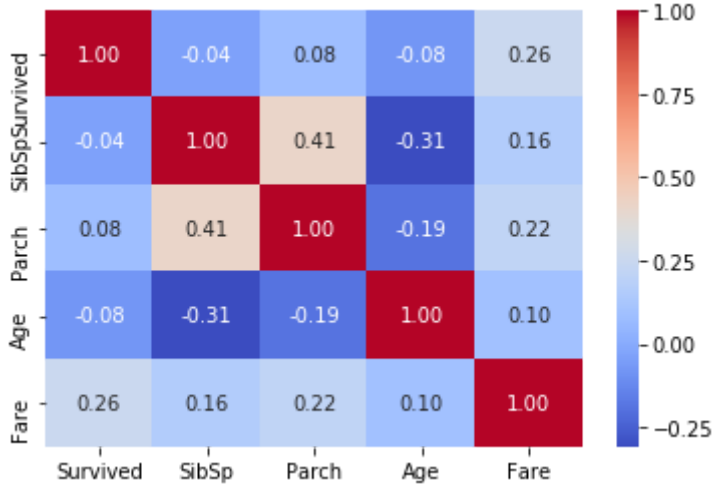
In [98]:
```
sns.swarmplot(x='Pclass',y='Age',data=df_train)
```

Out[98]: <matplotlib.axes._subplots.AxesSubplot at 0x1efd09a3e48>



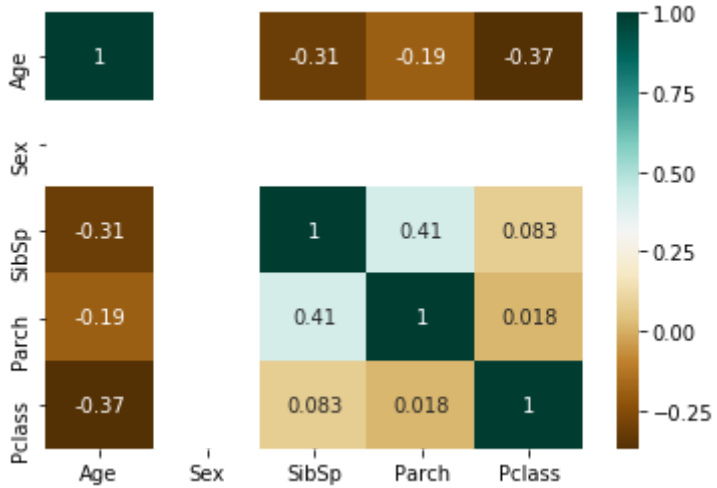
2-10 Heatmap

In [99]:
```
g = sns.heatmap(df_train[["Survived","SibSp","Parch","Age","Fare"]].corr(),
                annot=True, fmt = ".2f", cmap = "coolwarm")
```

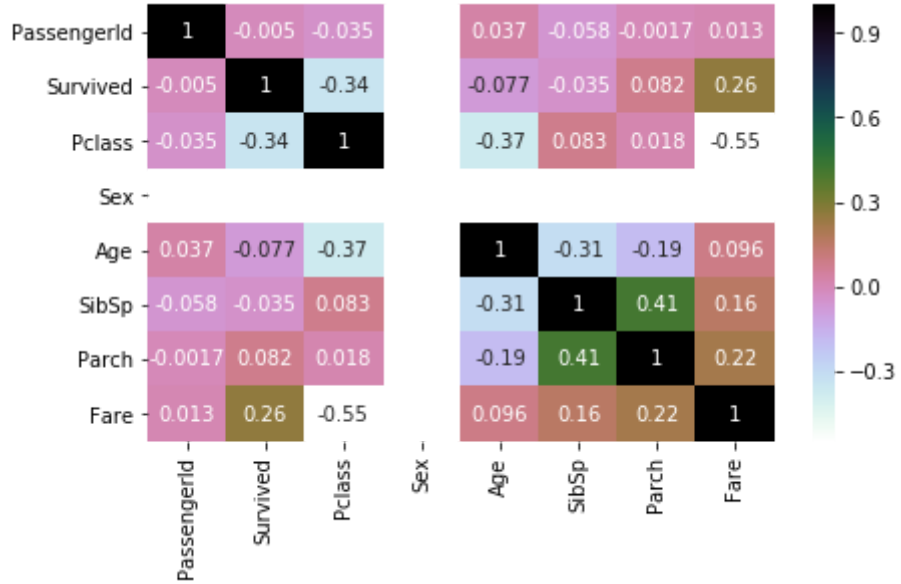


In [100]:
```
df_train["Sex"] = df_train["Sex"].map({"male": 0, "female":1})
```

In [101]:
```
g = sns.heatmap(df_train[["Age","Sex","SibSp","Parch","Pclass"]].corr(),cmap="BrBG",annot=True)
```



In [102]:
```
plt.figure(figsize=(7,4))
sns.heatmap(df_train.corr(),annot=True,cmap='cubehelix_r') # 상관관계를 Heatmap를 통해 표시합니다.
plt.show()
```



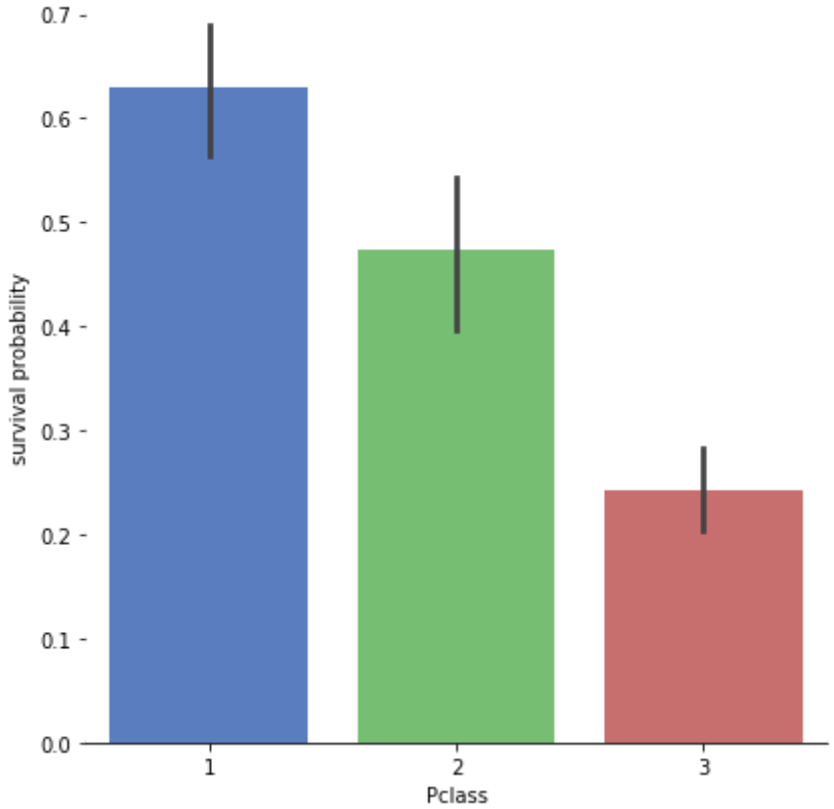
In [105]:
```
#correlation heatmap of dataset
def correlation_heatmap(df):
    _ , ax = plt.subplots(figsize =(14, 12))
    colormap = sns.diverging_palette(220, 10, as_cmap = True)

    _ = sns.heatmap(
        df.corr(),
        cmap = colormap,
        square=True,
        cbar_kws={'shrink':.9 },
        ax=ax,
        annot=True,
        linewidths=0.1,vmax=1.0, linecolor='white',
        annot_kws={'fontsize':12 }
    )

    plt.title('Pearson Correlation of Features', y=1.05, size=15)

correlation_heatmap(df_train)
```

2-11 Bar Plot

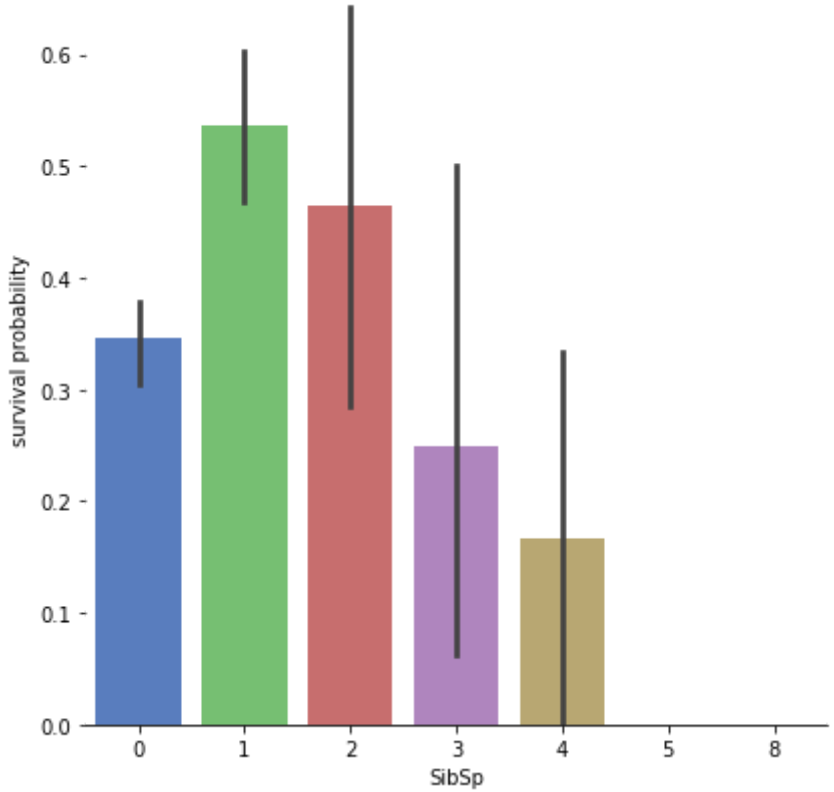
In [107]:
```
df_train.columns
```

Out[107]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

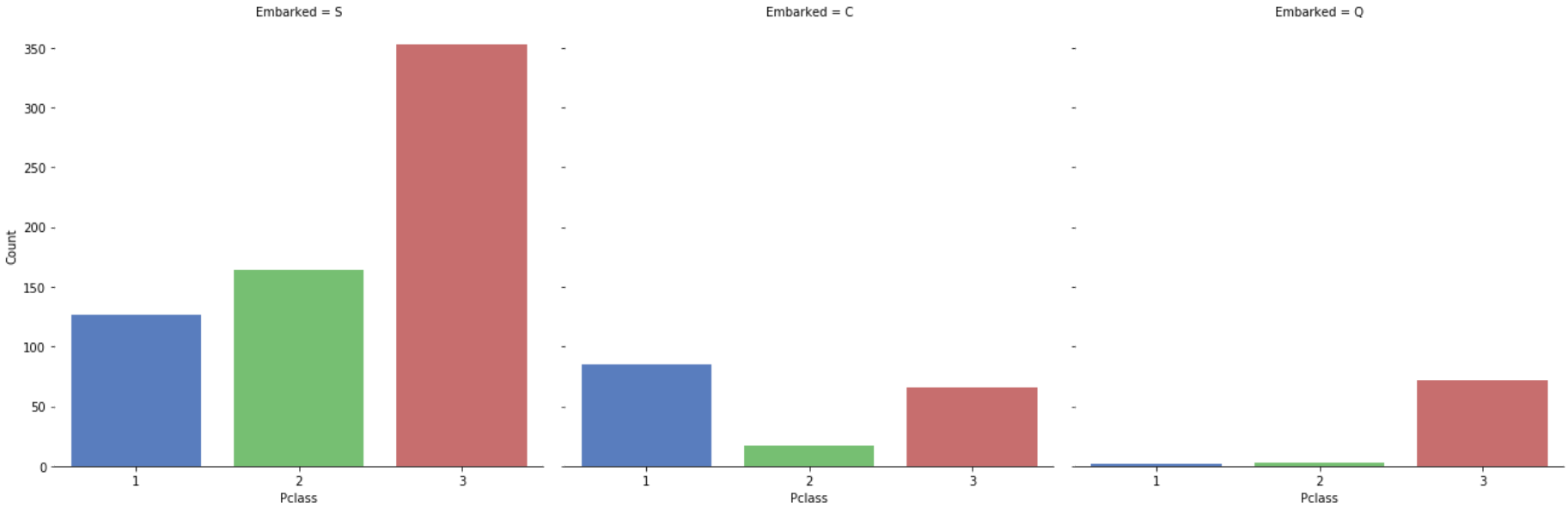
In [109]:
```
# Explore Pclass vs Survived
g = sns.factorplot(x="Pclass",y="Survived",data=df_train,kind="bar", size = 6 ,
palette = "muted")
g.despine(left=True)
g = g.set_ylabels("survival probability")
```



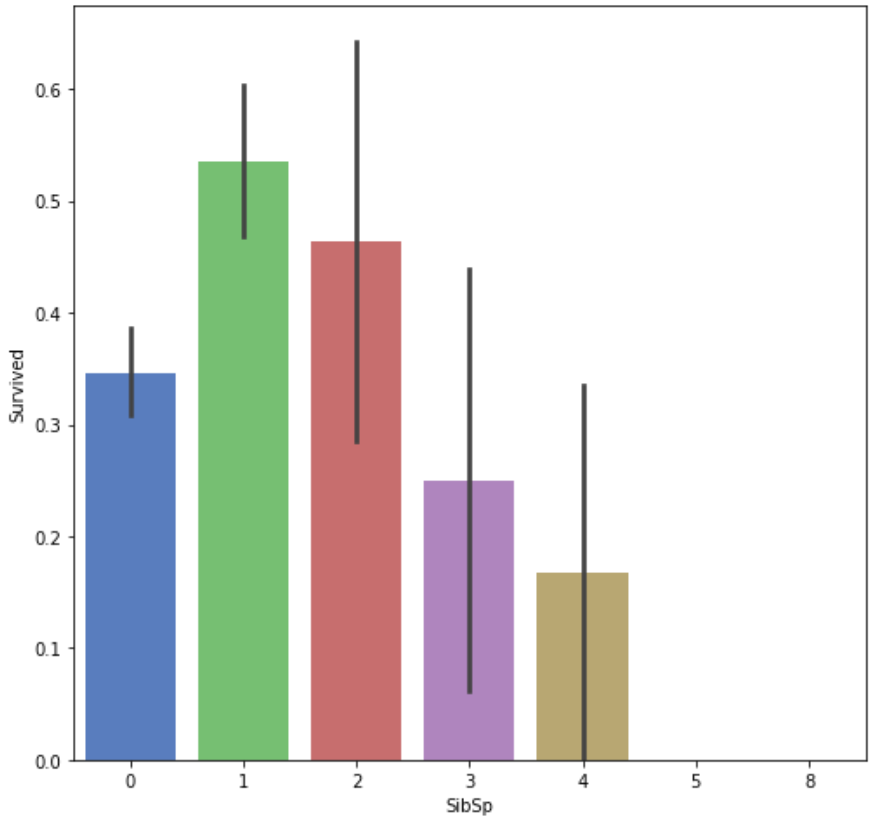
In [110]:
```
# Explore SibSp feature vs Survived
g = sns.factorplot(x="SibSp",y="Survived",
                   data=df_train,
                   kind="bar", size = 6 , palette = "muted")

g.despine(left=True)   # 왼쪽 선을 없애기(True, False)
g = g.set_ylabels("survival probability")
```

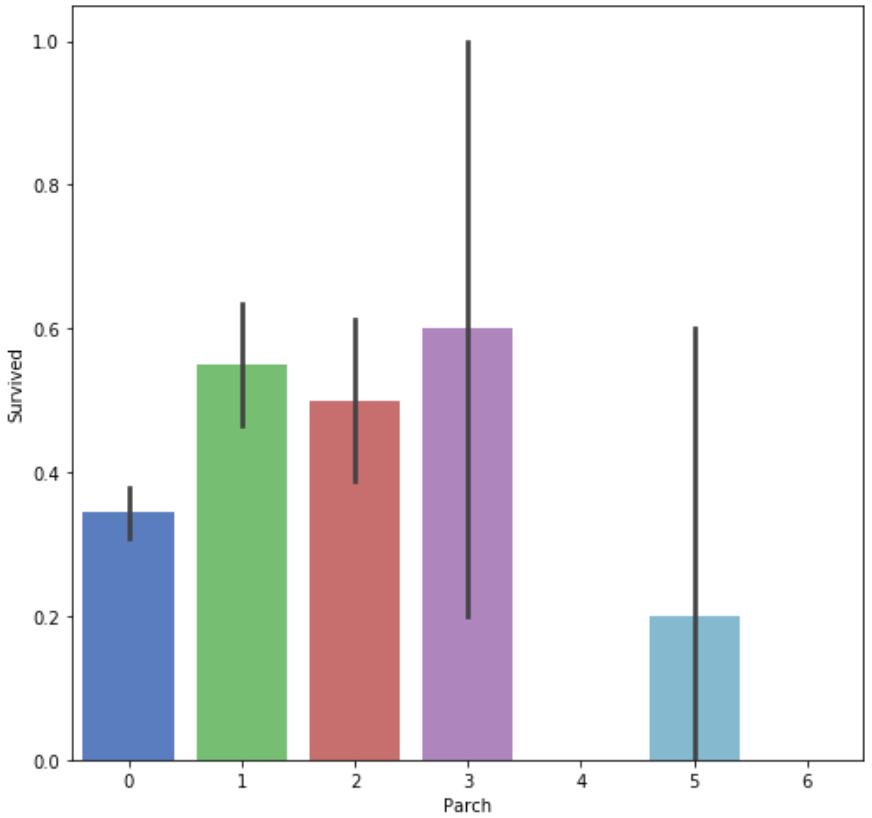


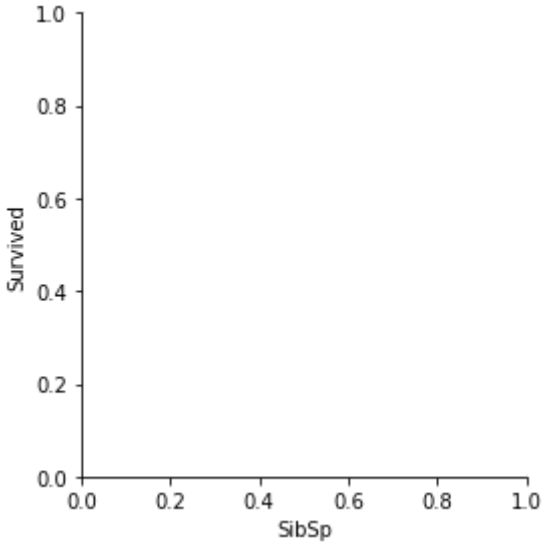
In [112]:
```
g = sns.factorplot("Pclass", col="Embarked",  data=df_train,
                   size=6, kind="count", palette="muted")
g.despine(left=True)
g = g.set_ylabels("Count")
```

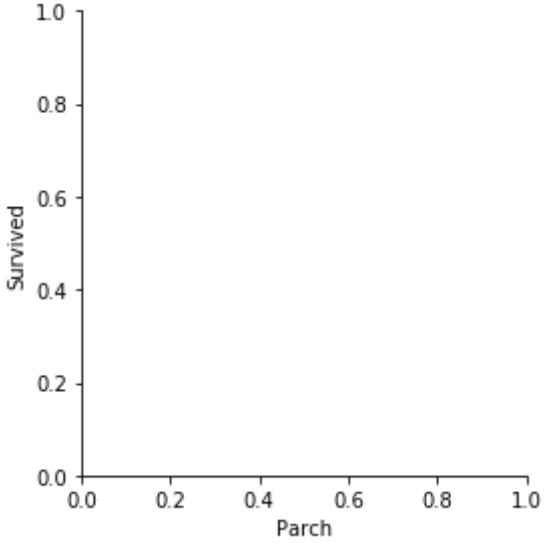


In [113]:
```
f,ax=plt.subplots(1,2,figsize=(18,8))

# Explore SibSp feature vs Survived
sns.factorplot(x="SibSp",y="Survived",
                   data=df_train,
                   kind="bar", palette = "muted", ax=ax[0])


# Explore Parch feature vs Survived
sns.factorplot(x="Parch",y="Survived",data=df_train,
                    kind="bar",palette = "muted", ax=ax[1])

plt.show()
```

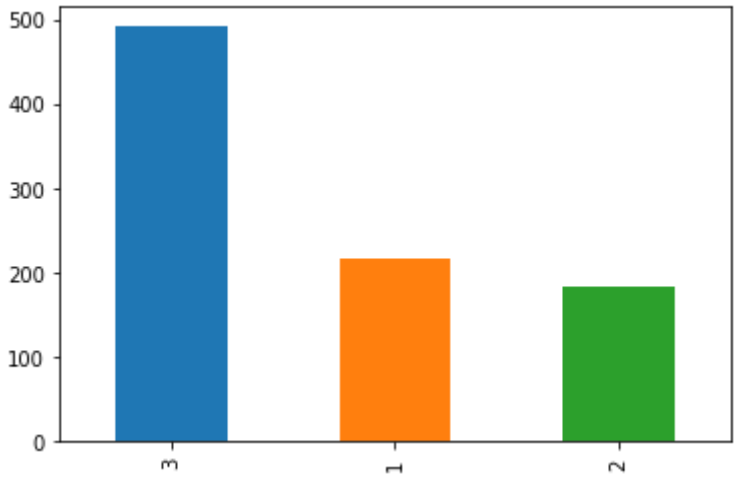




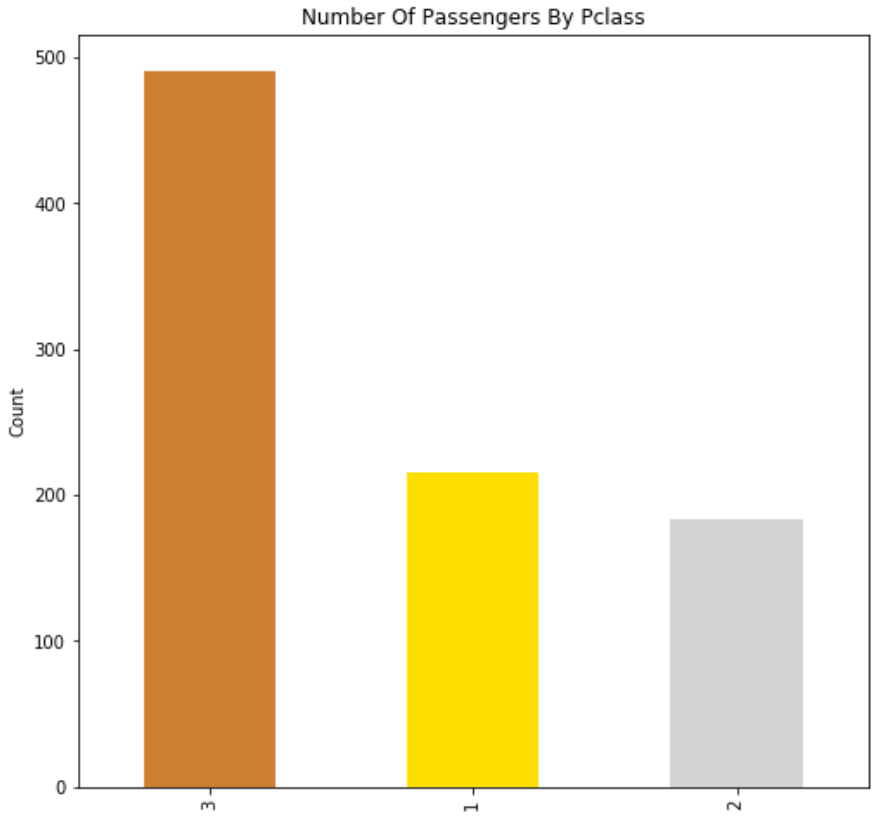

1.0
0.8
0.6
0.4
0.2
0.0
Survived
0.0
0.2
0.4
0.6
0.8
1.0
SibSp

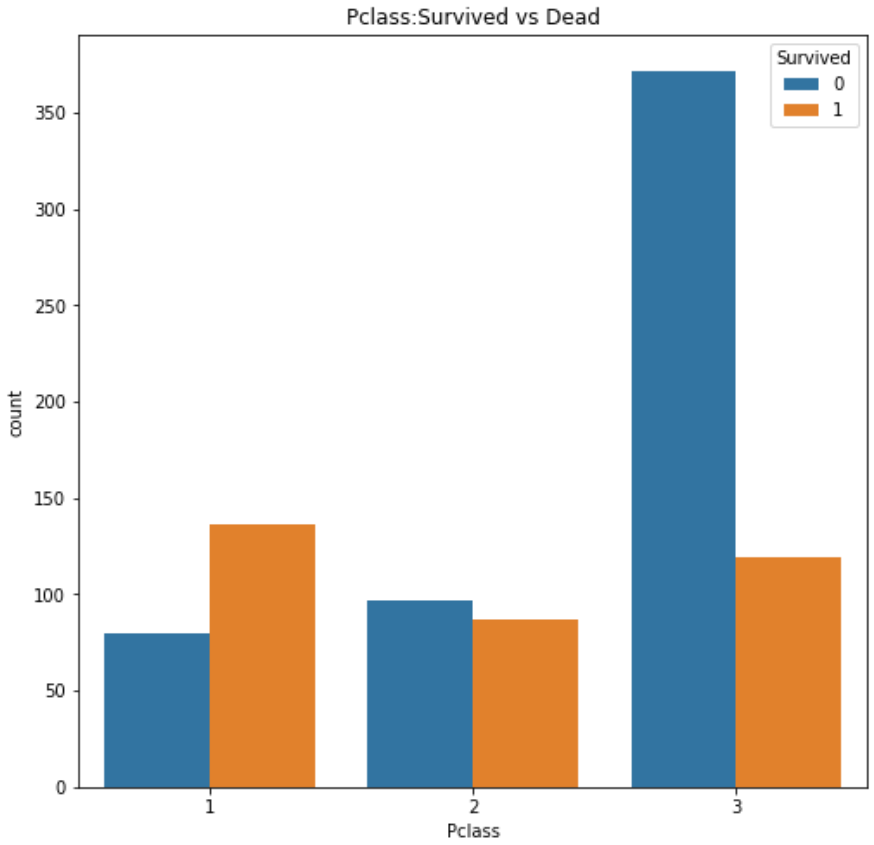

1.0
0.8
0.6
0.4
0.2
0.0
Survived
0.0
0.2
0.4
0.6
0.8
1.0
Parch

In [114]:
```
df_train['Pclass'].value_counts().plot(kind="bar");
```



In [138]:
```
f,ax=plt.subplots(1,2,figsize=(18,8))
df_train['Pclass'].value_counts().plot.bar(color=['#CD7F32','#FFDF00','#D3D3D3'],ax=ax[0])
ax[0].set_title('Number Of Passengers By Pclass')
ax[0].set_ylabel('Count')
sns.countplot('Pclass',hue='Survived',data=df_train,ax=ax[1])
ax[1].set_title('Pclass:Survived vs Dead')
plt.show()
```





In [115]:
```
train_df.columns
```

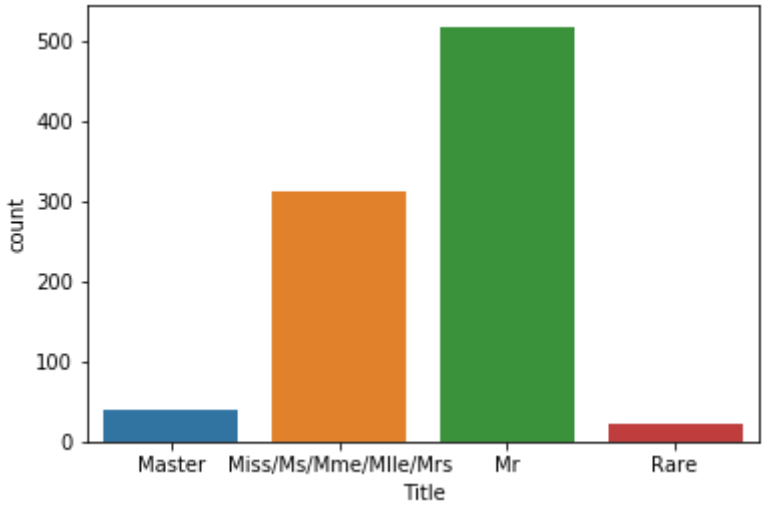
Out[115]:
```
Index(['Survived', 'Pclass', 'Age', 'Fare', 'Cabin', 'C', 'Q', 'Family',
       'Child', 'Female'],
      dtype='object')
```

In [116]:
```
# Get Title from Name
dataset_title = [i.split(",")[1].split(".")[0].strip() for i in df_train["Name"]]
df_train["Title"] = pd.Series(dataset_title)
df_train["Title"].head()
```

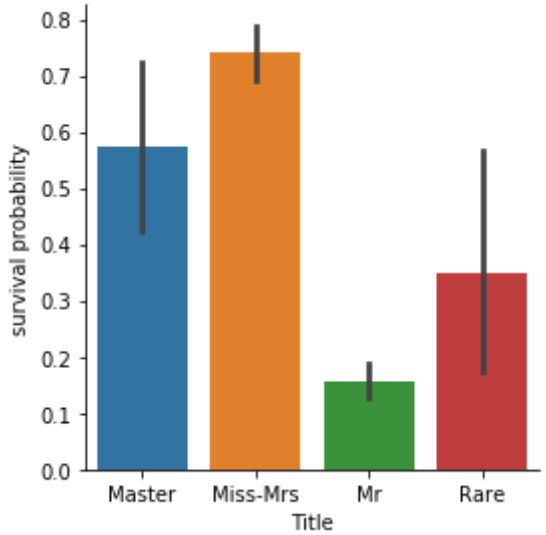
Out[116]:
```
0      Mr
1     Mrs
2    Miss
3     Mrs
4      Mr
Name: Title, dtype: object
```

In [117]:
```
# Convert to categorical values Title
df_train["Title"] = df_train["Title"].replace(['Lady', 'the Countess',
                                               'Countess','Capt',
                                               'Col','Don', 'Dr',
                                               'Major', 'Rev', 'Sir', 'Jonkheer', 'Dona'], 'Rare')
df_train["Title"] = df_train["Title"].map({"Master":0, "Miss":1,
                                           "Ms" : 1 , "Mme":1, "Mlle":1, "Mrs":1, "Mr":2, "Rare":3})
df_train["Title"] = df_train["Title"].astype(int)
```

In [119]:
```
g = sns.countplot(df_train["Title"])
g = g.set_xticklabels(["Master","Miss/Ms/Mme/Mlle/Mrs","Mr","Rare"])
```



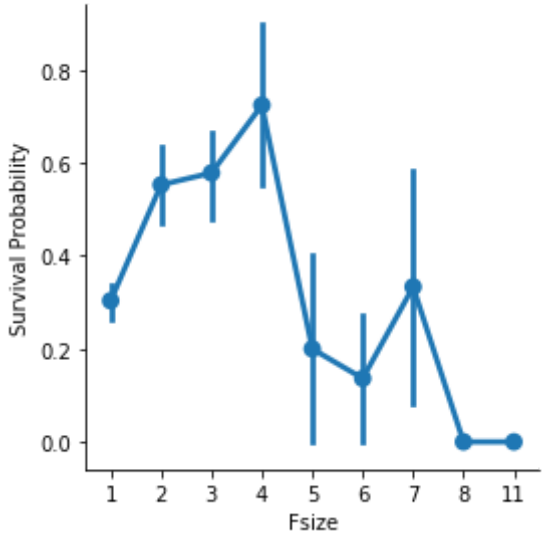
In [121]:
```
g = sns.factorplot(x="Title",y="Survived",data=df_train,kind="bar")
g = g.set_xticklabels(["Master","Miss-Mrs","Mr","Rare"])
g = g.set_ylabels("survival probability")
```



Family Size

우리는 대가족이 피난하는 동안 자매 / 형제 / 부모를 찾고 대피하는 것이 더 어려울 것이라고 생각할 수 있습니다. 그래서 SibSp, Parch와 1 (승객 포함)의 합계 인 "Fize"(가족 크기) 기능을 만들도록 선택했습니다.

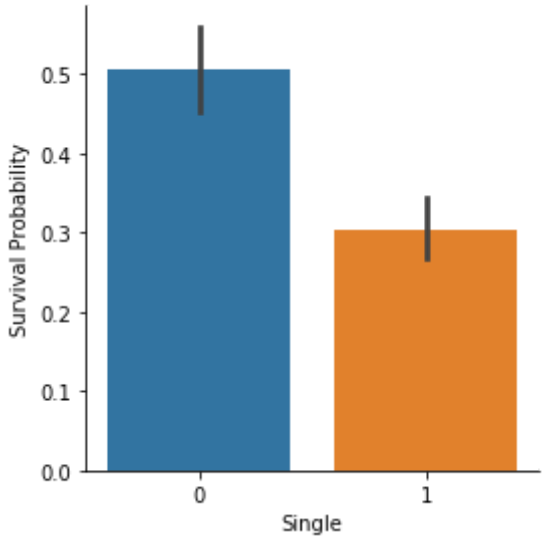
In [123]:
```
# Create a family size descriptor from SibSp and Parch
df_train["Fsize"] = df_train["SibSp"] + df_train["Parch"] + 1

g = sns.factorplot(x="Fsize",y="Survived",data = df_train)
g = g.set_ylabels("Survival Probability")
```

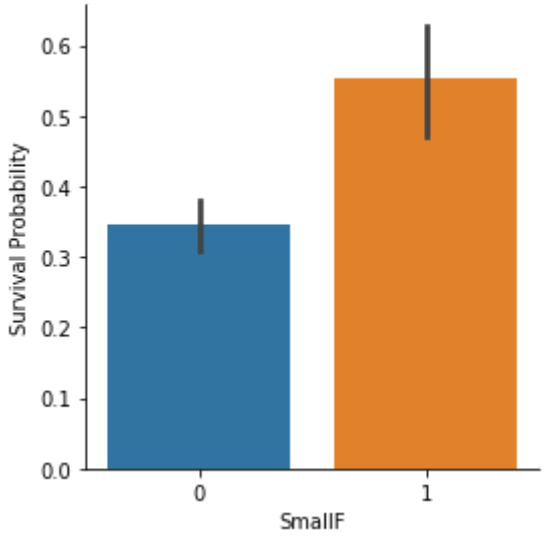


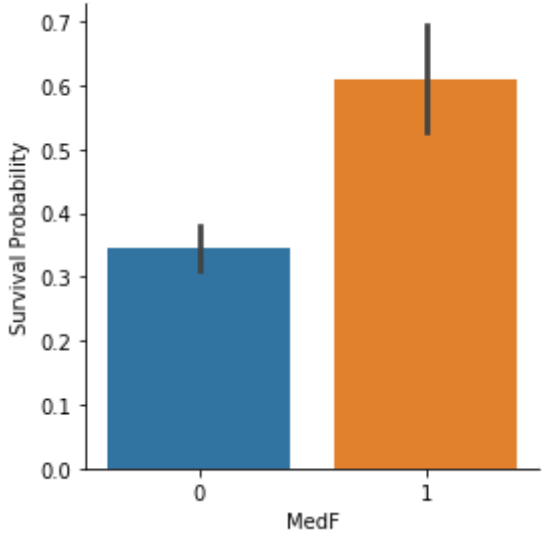
In [125]:
```
# Create new feature of family size
df_train['Single'] = df_train['Fsize'].map(lambda s: 1 if s == 1 else 0)
df_train['SmallF'] = df_train['Fsize'].map(lambda s: 1 if  s == 2  else 0)
df_train['MedF'] = df_train['Fsize'].map(lambda s: 1 if 3 <= s <= 4 else 0)
df_train['LargeF'] = df_train['Fsize'].map(lambda s: 1 if s >= 5 else 0)
```

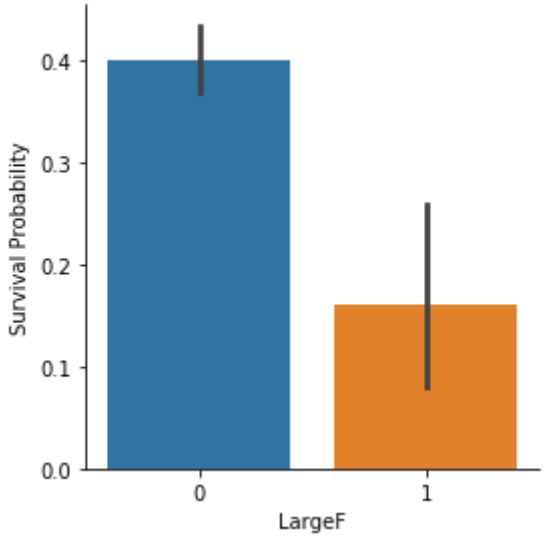
In [ ]:
```
```

In [135]:
```
g = sns.factorplot(x="Single",y="Survived",data=df_train,kind="bar")
g = g.set_ylabels("Survival Probability")
g = sns.factorplot(x="SmallF",y="Survived",data=df_train,kind="bar")
g = g.set_ylabels("Survival Probability")
g = sns.factorplot(x="MedF",y="Survived",data=df_train,kind="bar")
g = g.set_ylabels("Survival Probability")
g = sns.factorplot(x="LargeF",y="Survived",data=df_train,kind="bar")
g = g.set_ylabels("Survival Probability")
```

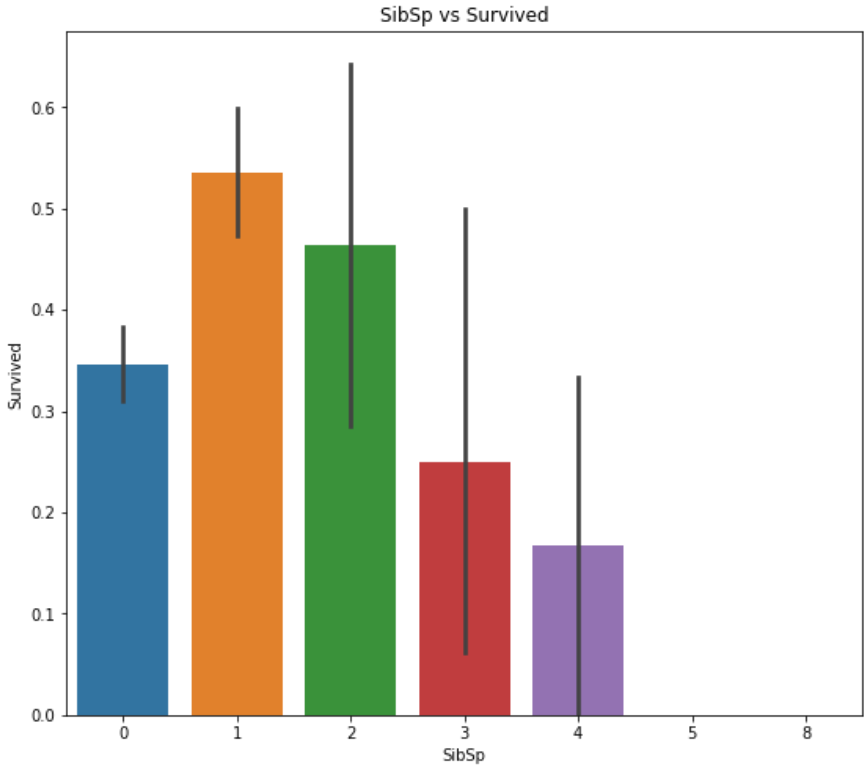


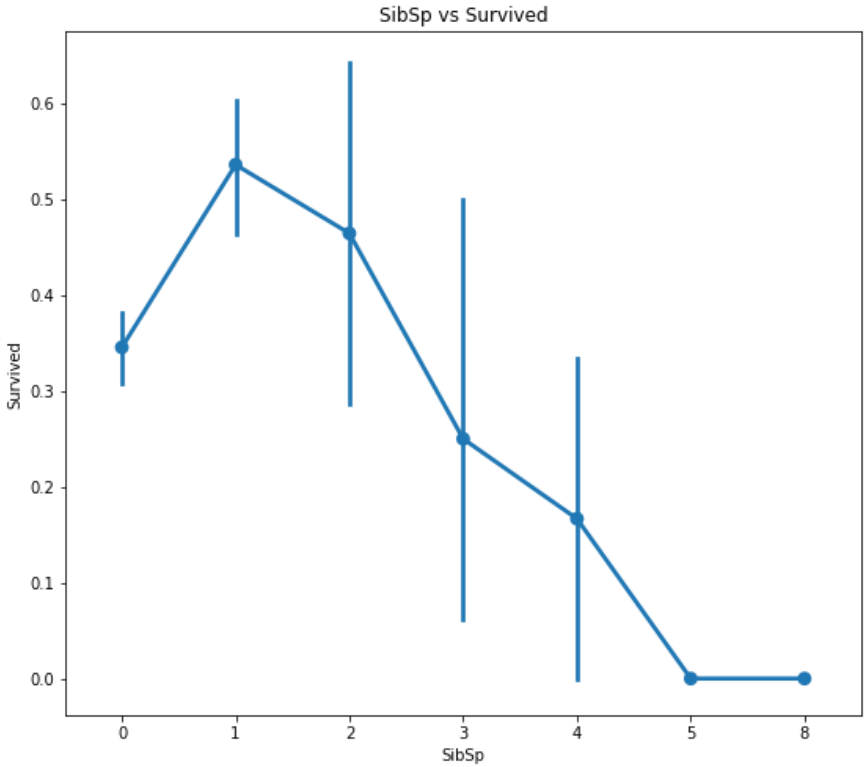

Survival Probability
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
1
SmallF


Survival Probability
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
1
MedF


0.4
0.3
0.2
0.1
0.0
Survival Probability
0
1
LargeF

In [140]:
```
f,ax=plt.subplots(1,2,figsize=(20,8))
sns.barplot('SibSp','Survived',data=df_train,ax=ax[0])
ax[0].set_title('SibSp vs Survived')
sns.factorplot('SibSp','Survived',data=df_train,ax=ax[1])
ax[1].set_title('SibSp vs Survived')
plt.close(2)
plt.show()
```





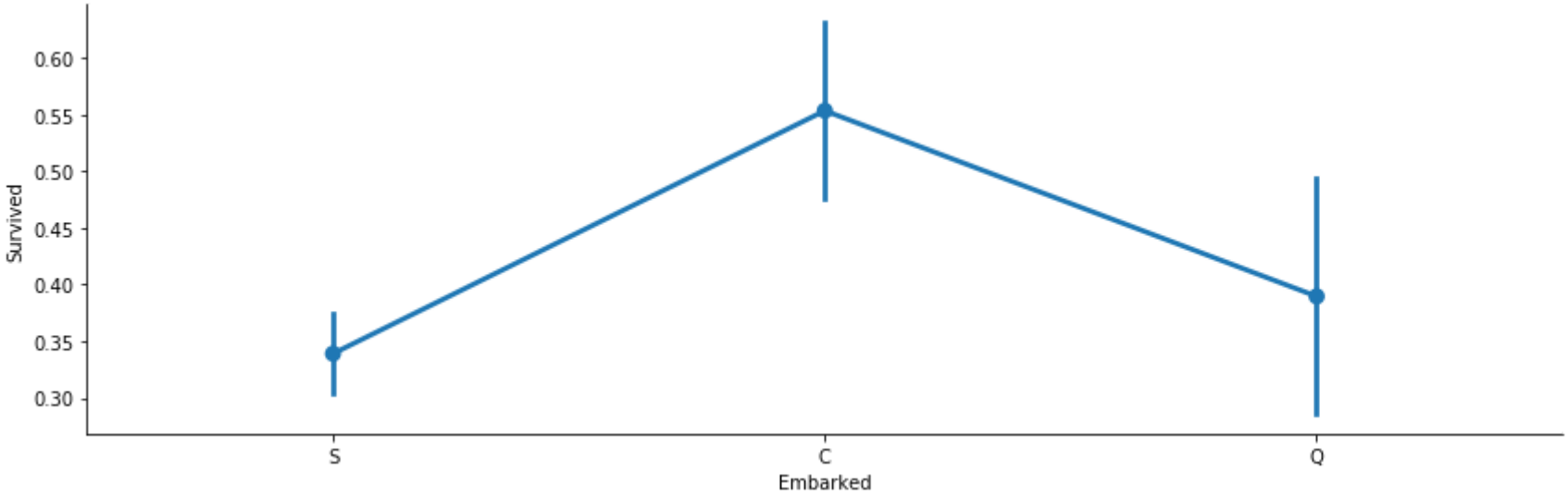
In [26]:
```
# get titanic & test csv files as a DataFrame
train_df = pd.read_csv("data/train.csv")
test_df    = pd.read_csv("data/test.csv")

# preview the data
train_df.head()
```

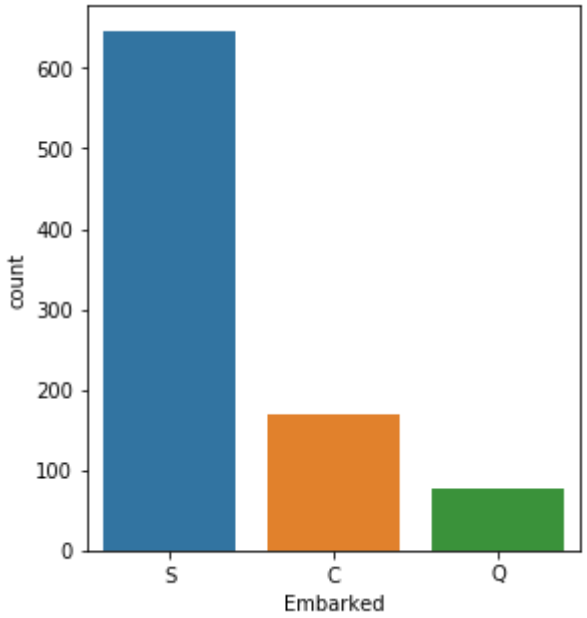
Out[26]:

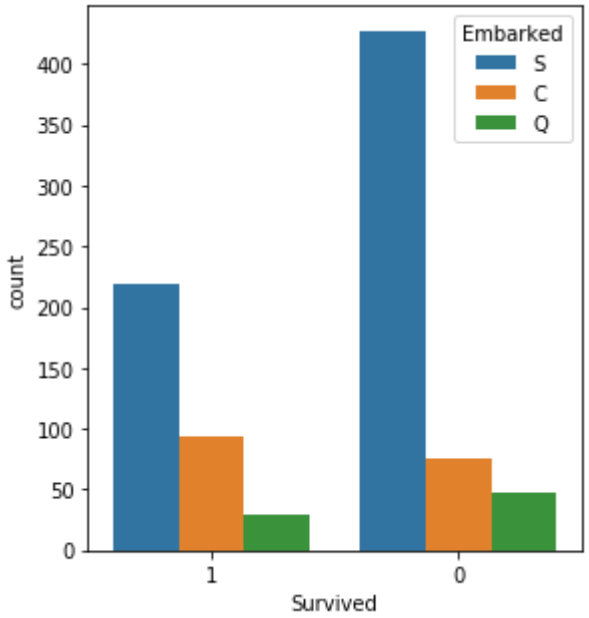
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 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

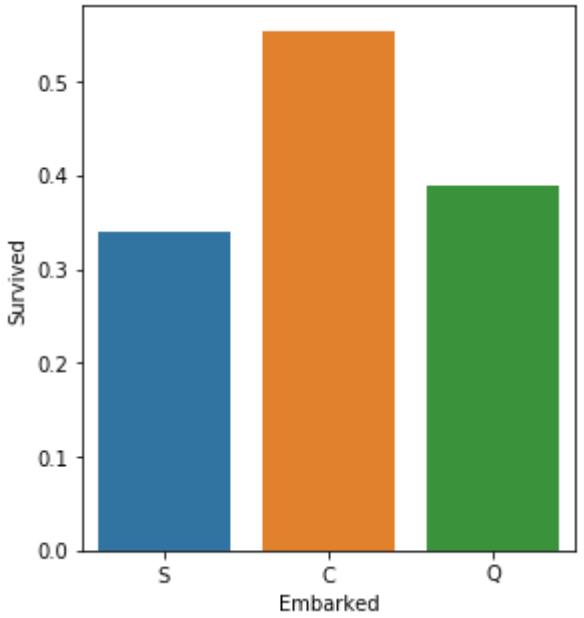
In [27]:
```
# drop unnecessary columns, these columns won't be useful in analysis and prediction
train_df = train_df.drop(['PassengerId','Name','Ticket'], axis=1)
test_df = test_df.drop(['Name','Ticket'], axis=1)
```

In [28]:
```
# Embarked

# only in titanic_df, fill the two missing values with the most occurred value, which is "S".
train_df["Embarked"] = train_df["Embarked"].fillna("S")

# plot
sns.factorplot('Embarked','Survived', data=train_df,size=4,aspect=3)

fig, (axis1,axis2,axis3) = plt.subplots(1,3,figsize=(15,5))

# sns.factorplot('Embarked',data=titanic_df,kind='count',order=['S','C','Q'],ax=axis1)
# sns.factorplot('Survived',hue="Embarked",data=titanic_df,kind='count',order=[1,0],ax=axis2)
sns.countplot(x='Embarked', data=train_df, ax=axis1)
sns.countplot(x='Survived', hue="Embarked", data=train_df, order=[1,0], ax=axis2)

# group by embarked, and get the mean for survived passengers for each value in Embarked
embark_perc = train_df[["Embarked", "Survived"]].groupby(['Embarked'],as_index=False).mean()
sns.barplot(x='Embarked', y='Survived', data=embark_perc,order=['S','C','Q'],ax=axis3)

# Either to consider Embarked column in predictions,
# and remove "S" dummy variable,
# and leave "C" & "Q", since they seem to have a good rate for Survival.

# OR, don't create dummy variables for Embarked column, just drop it,
# because logically, Embarked doesn't seem to be useful in prediction.

embark_dummies_titanic  = pd.get_dummies(train_df['Embarked'])
embark_dummies_titanic.drop(['S'], axis=1, inplace=True)

embark_dummies_test  = pd.get_dummies(test_df['Embarked'])
embark_dummies_test.drop(['S'], axis=1, inplace=True)

train_df = train_df.join(embark_dummies_titanic)
test_df    = test_df.join(embark_dummies_test)

train_df.drop(['Embarked'], axis=1,inplace=True)
test_df.drop(['Embarked'], axis=1,inplace=True)
```


Survived
0.60
0.55
0.50
0.45
0.40
0.35
0.30
S
C
Q
Embarked


count
600
500
400
300
200
100
0
S
C
Q
Embarked

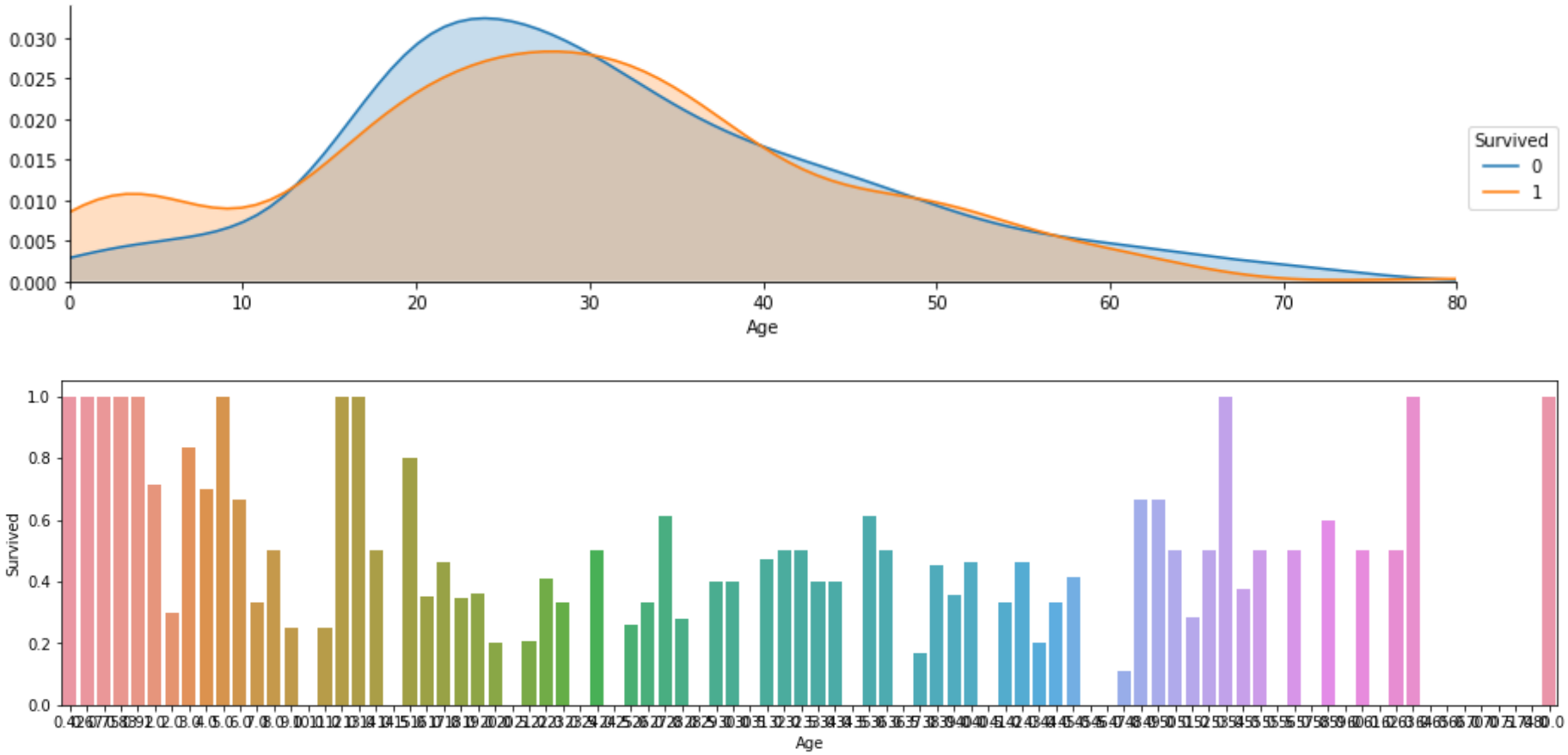

count
400
350
300
250
200
150
100
50
0
1
0
Survived
Embarked
S
C
Q


Survived
0.5
0.4
0.3
0.2
0.1
0.0
S
C
Q
Embarked

In [ ]:

In [29]:
```
# peaks for survived/not survived passengers by their age
facet = sns.FacetGrid(train_df, hue="Survived",aspect=4)
facet.map(sns.kdeplot,'Age',shade= True)
facet.set(xlim=(0, train_df['Age'].max()))
facet.add_legend()

# average survived passengers by age
fig, axis1 = plt.subplots(1,1,figsize=(18,4))
average_age = train_df[["Age", "Survived"]].groupby(['Age'],as_index=False).mean()
sns.barplot(x='Age', y='Survived', data=average_age)
```

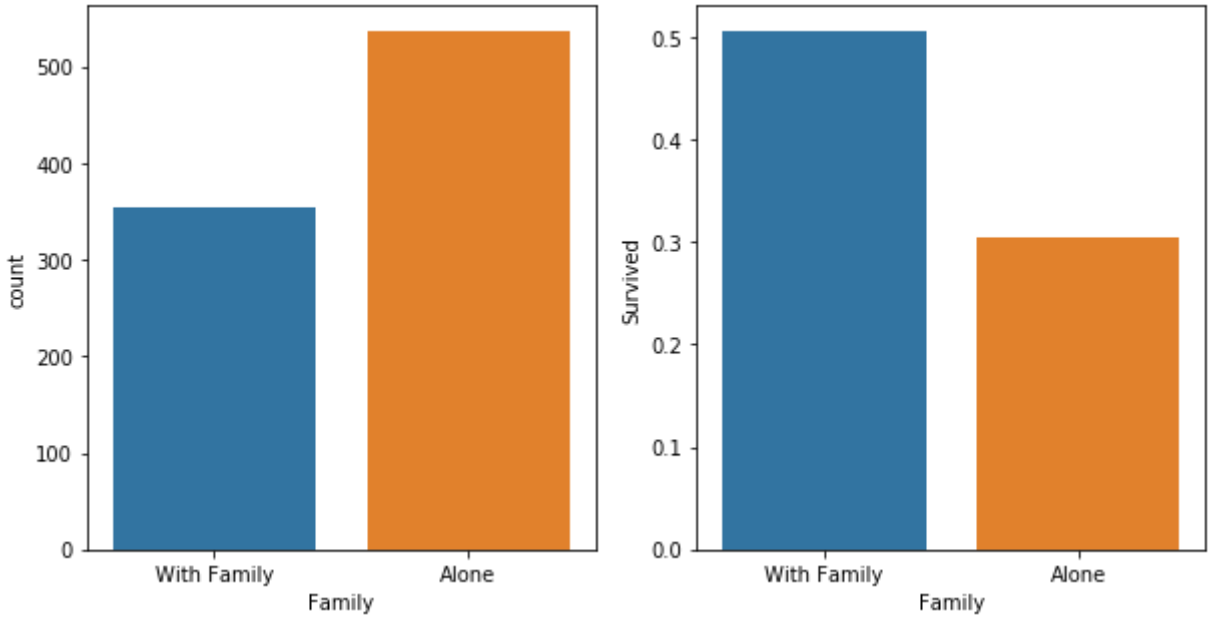
Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x1efcf840c18>



In [30]:
```
# Family

# Instead of having two columns Parch & SibSp,
# we can have only one column represent if the passenger had any family member aboard or not,
# Meaning, if having any family member(whether parent, brother, ...etc) will increase chances of Survival or not.
train_df['Family'] =  train_df["Parch"] + train_df["SibSp"]
train_df['Family'].loc[train_df['Family'] > 0] = 1
train_df['Family'].loc[train_df['Family'] == 0] = 0

test_df['Family'] =  test_df["Parch"] + test_df["SibSp"]
test_df['Family'].loc[test_df['Family'] > 0] = 1
test_df['Family'].loc[test_df['Family'] == 0] = 0

# drop Parch & SibSp
train_df = train_df.drop(['SibSp','Parch'], axis=1)
test_df    = test_df.drop(['SibSp','Parch'], axis=1)

# plot
fig, (axis1,axis2) = plt.subplots(1,2,sharex=True,figsize=(10,5))

# sns.factorplot('Family',data=titanic_df,kind='count',ax=axis1)
sns.countplot(x='Family', data=train_df, order=[1,0], ax=axis1)

# average of survived for those who had/didn't have any family member
family_perc = train_df[["Family", "Survived"]].groupby(['Family'],as_index=False).mean()
sns.barplot(x='Family', y='Survived', data=family_perc, order=[1,0], ax=axis2)

axis1.set_xticklabels(["With Family","Alone"], rotation=0)
```

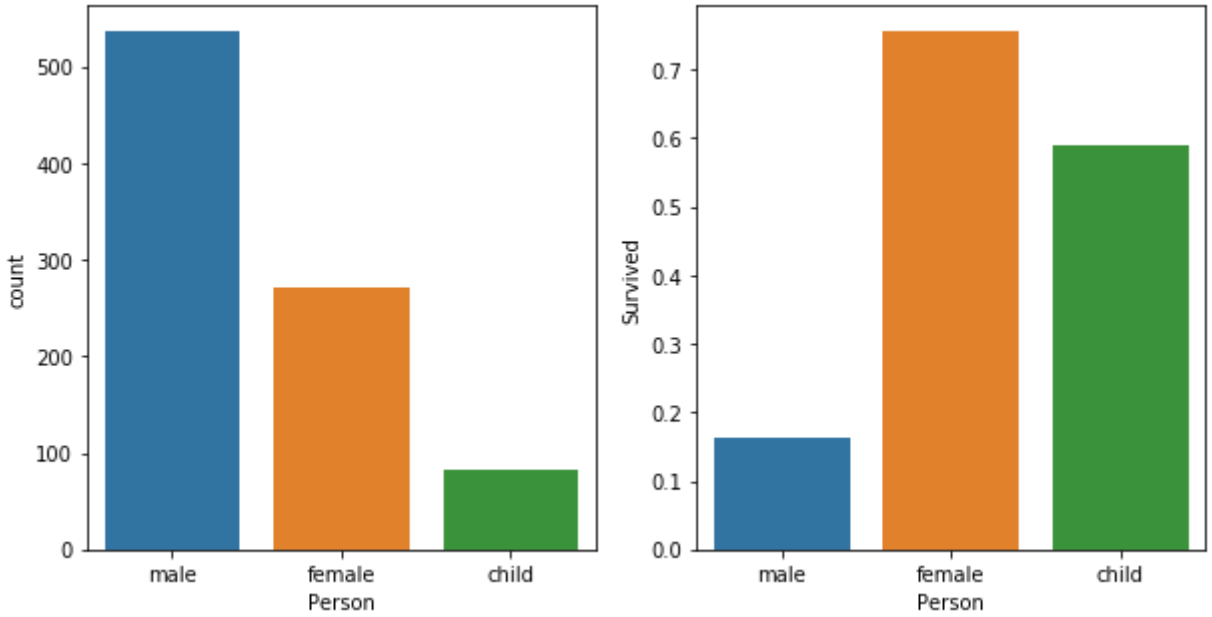
```
C:₩Users₩WITHJS₩Anaconda3₩lib₩site-packages₩pandas₩core₩indexing.py:194: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy)
  self._setitem_with_indexer(indexer, value)
```

Out[30]:
```
[Text(0,0,'With Family'), Text(0,0,'Alone')]
```


500
400
300
200
100
0
count
With Family
Alone
Family
0.5
0.4
0.3
0.2
0.1
0.0
Survived
With Family
Alone
Family

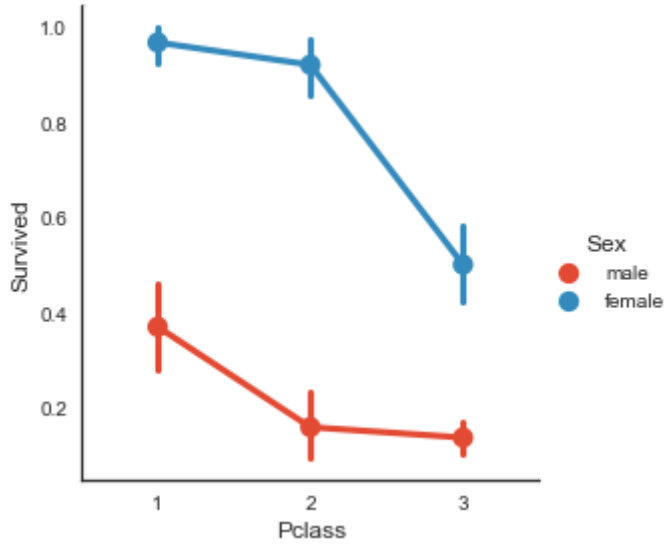
In [ ]:

In [31]:
```
# Sex

# As we see, children(age < ~16) on aboard seem to have a high chances for Survival.
# So, we can classify passengers as males, females, and child
def get_person(passenger):
    age,sex = passenger
    return 'child' if age < 16 else sex

train_df['Person'] = train_df[['Age','Sex']].apply(get_person,axis=1)
test_df['Person']    = test_df[['Age','Sex']].apply(get_person,axis=1)

# No need to use Sex column since we created Person column
train_df.drop(['Sex'],axis=1,inplace=True)
test_df.drop(['Sex'],axis=1,inplace=True)

# create dummy variables for Person column, & drop Male as it has the lowest average of survived passengers
person_dummies_titanic  = pd.get_dummies(train_df['Person'])
person_dummies_titanic.columns = ['Child','Female','Male']
person_dummies_titanic.drop(['Male'], axis=1, inplace=True)

person_dummies_test  = pd.get_dummies(test_df['Person'])
person_dummies_test.columns = ['Child','Female','Male']
person_dummies_test.drop(['Male'], axis=1, inplace=True)

train_df = train_df.join(person_dummies_titanic)
test_df    = test_df.join(person_dummies_test)

fig, (axis1,axis2) = plt.subplots(1,2,figsize=(10,5))

# sns.factorplot('Person',data=titanic_df,kind='count',ax=axis1)
sns.countplot(x='Person', data=train_df, ax=axis1)

# average of survived for each Person(male, female, or child)
person_perc = train_df[["Person", "Survived"]].groupby(['Person'],as_index=False).mean()
sns.barplot(x='Person', y='Survived', data=person_perc, ax=axis2, order=['male','female','child'])

train_df.drop(['Person'],axis=1,inplace=True)
test_df.drop(['Person'],axis=1,inplace=True)
```



2-12 Factorplot

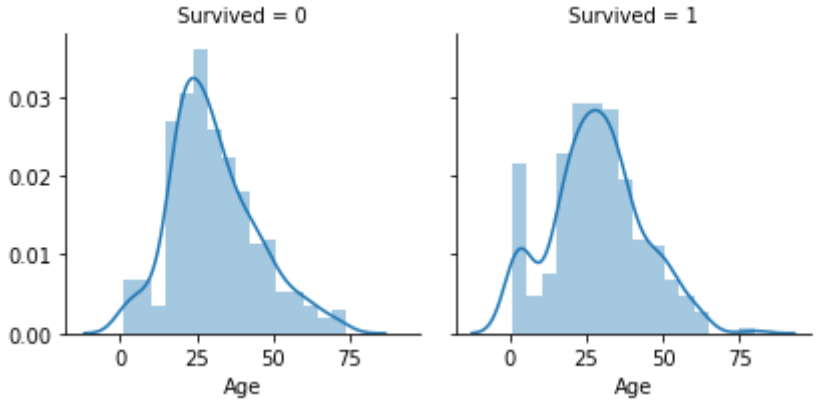
In [61]:
```
sns.factorplot('Pclass','Survived',hue='Sex',data=df_train)
plt.show()
```



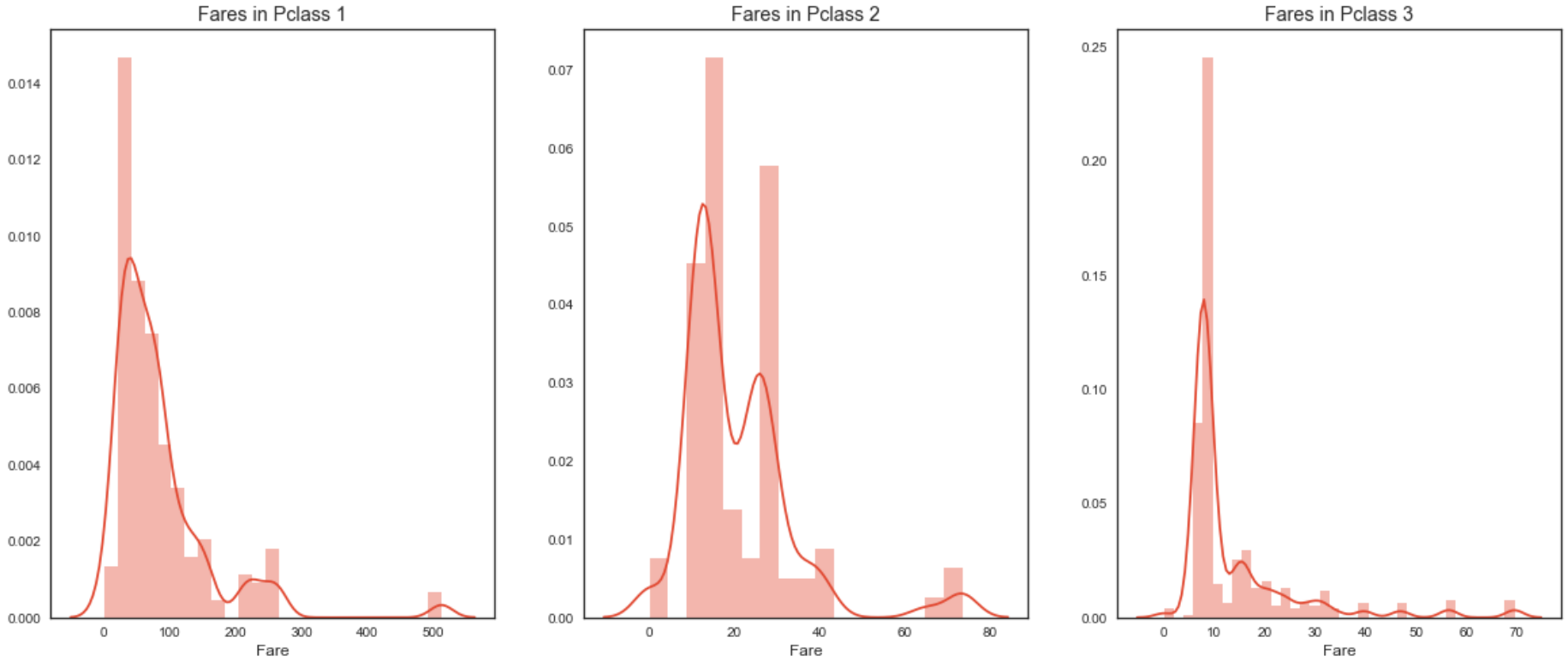
2-13 distplot

In [50]:

```
# Explore Age vs Survived
g = sns.FacetGrid(df_train, col='Survived')
g = g.map(sns.distplot, "Age")
```



In [63]:
```
f,ax=plt.subplots(1,3,figsize=(20,8))
sns.distplot(df_train[df_train['Pclass']==1].Fare,ax=ax[0])
ax[0].set_title('Fares in Pclass 1')
sns.distplot(df_train[df_train['Pclass']==2].Fare,ax=ax[1])
ax[1].set_title('Fares in Pclass 2')
sns.distplot(df_train[df_train['Pclass']==3].Fare,ax=ax[2])
ax[2].set_title('Fares in Pclass 3')
plt.show()
```



Ref

https://www.kaggle.com/mjbahmani/a-comprehensive-ml-workflow-with-python (https://www.kaggle.com/mjbahmani/a-comprehensive-ml-workflow-with-python)
https://www.kaggle.com/ldfreeman3/a-data-science-framework-to-achieve-99-accuracy (https://www.kaggle.com/ldfreeman3/a-data-science-

framework-to-achieve-99-accuracy)
https://www.kaggle.com/yassineghouzam/titanic-top-4-with-ensemble-modeling (https://www.kaggle.com/yassineghouzam/titanic-top-4-with-ensemble-modeling)

In [ ]: