In [1]:

```
#import libraries
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
```

In [2]:

```
test_df = pd.read_csv("data//test.csv")
train_df = pd.read_csv("data//train.csv")
```

In [3]:

```
test_df.head()
```

Out[3]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Emba
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	

In [4]:

```
train_df.head()
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	C
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	

In [5]:

```
train_df.describe()
```

Out[5]:

	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

In [6]:

```
train_df.shape
```

Out[6]:

```
(891, 12)
```

In [7]:

```
test_df.shape
```

Out[7]:

```
(418, 11)
```

In [8]:

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Name           891 non-null object
Sex            891 non-null object
Age            714 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

In [9]:

```
test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
PassengerId    418 non-null int64
Pclass         418 non-null int64
Name           418 non-null object
Sex            418 non-null object
Age            332 non-null float64
SibSp          418 non-null int64
Parch          418 non-null int64
Ticket         418 non-null object
Fare           417 non-null float64
Cabin          91 non-null object
Embarked       418 non-null object
dtypes: float64(2), int64(4), object(5)
memory usage: 36.0+ KB
```

In [10]:

```
train_df.isnull().sum()
```

Out[10]:

```
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 [11]:

```
test_df.isnull().sum()
```

Out[11]:

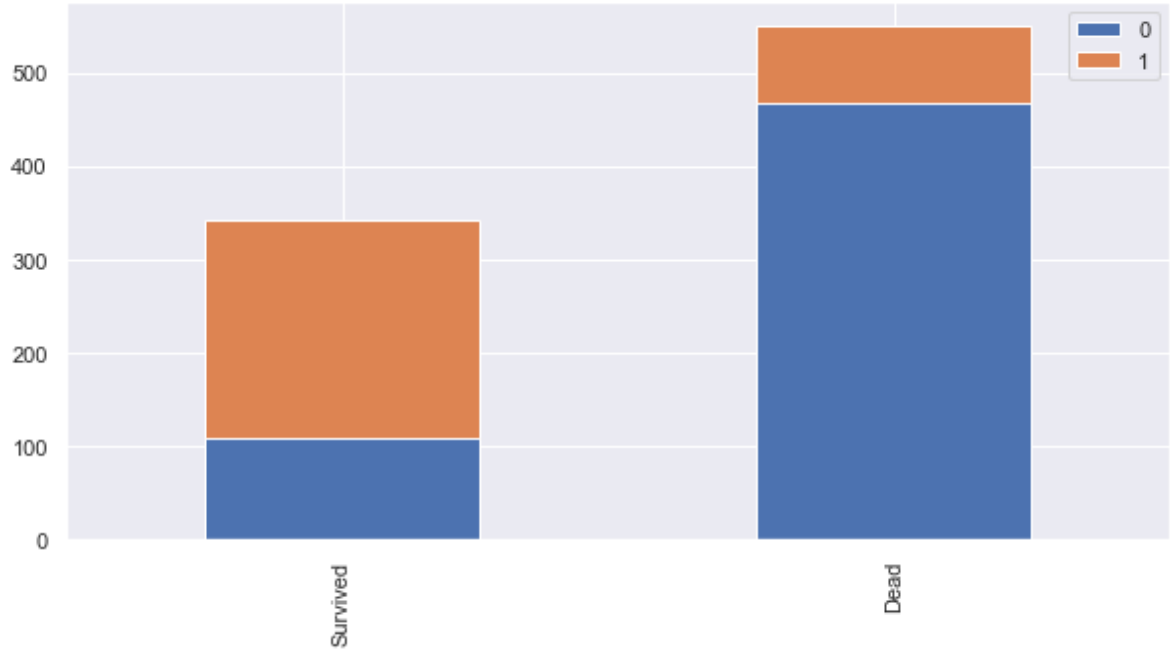
```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

In [121]:

```
def bar_chart(feature):
    survived = train_df[train_df['Survived']==1][feature].value_counts()
    dead = train_df[train_df['Survived']==0][feature].value_counts()
    df = pd.DataFrame([survived,dead])
    df.index = ['Survived','Dead']
    df.plot(kind='bar',stacked=True, figsize=(10,5))
```

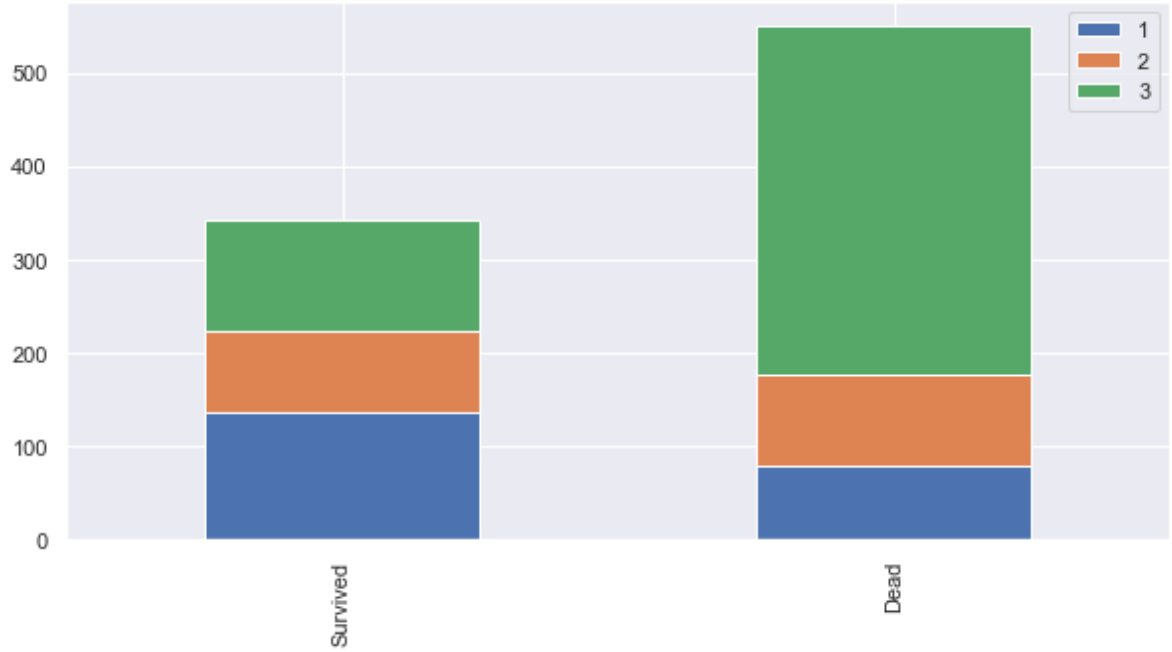
In [122]:

```
bar_chart("Sex")
```



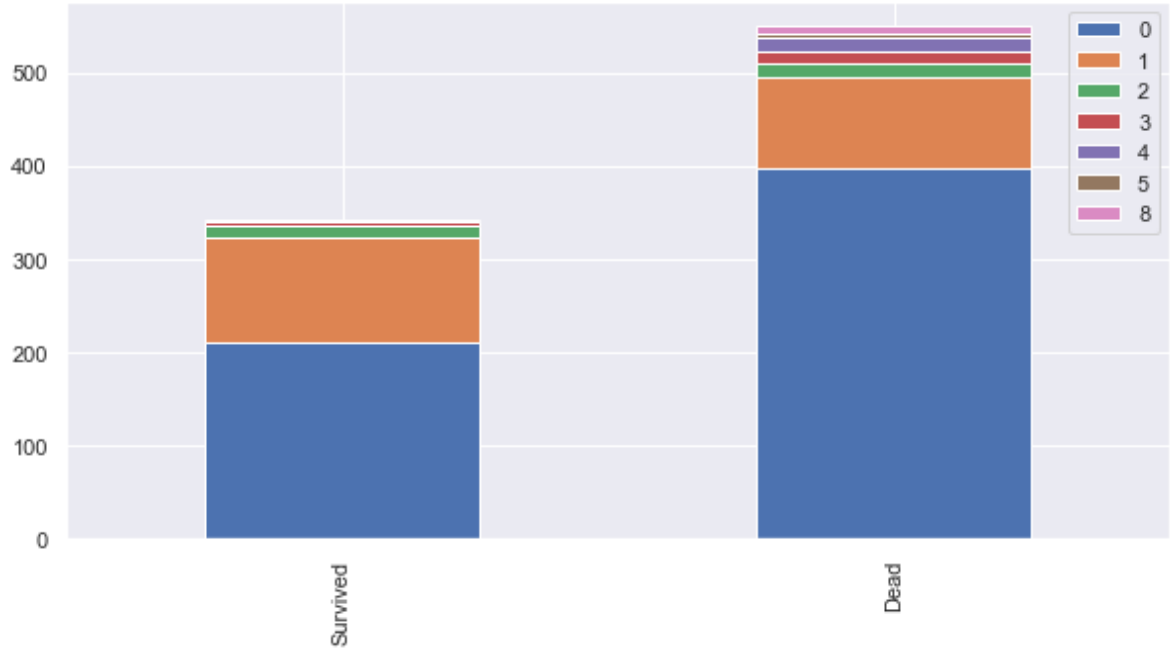
In [23]:

```
bar_chart("Pclass")
```



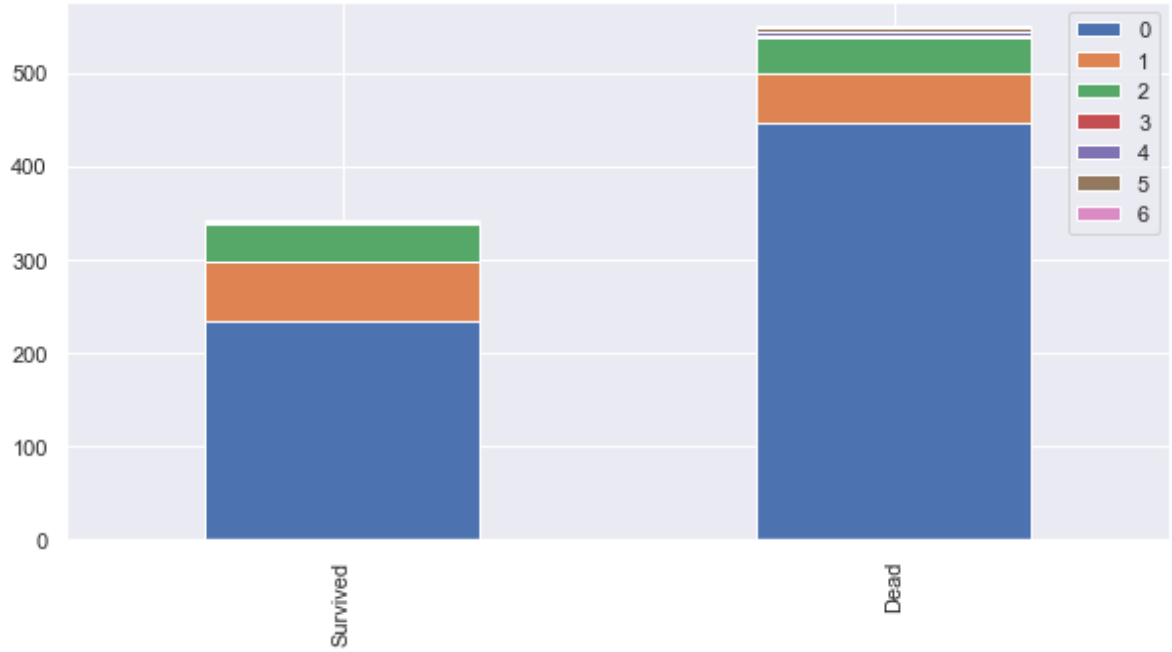
In [24]:

```
bar_chart("SibSp")
```



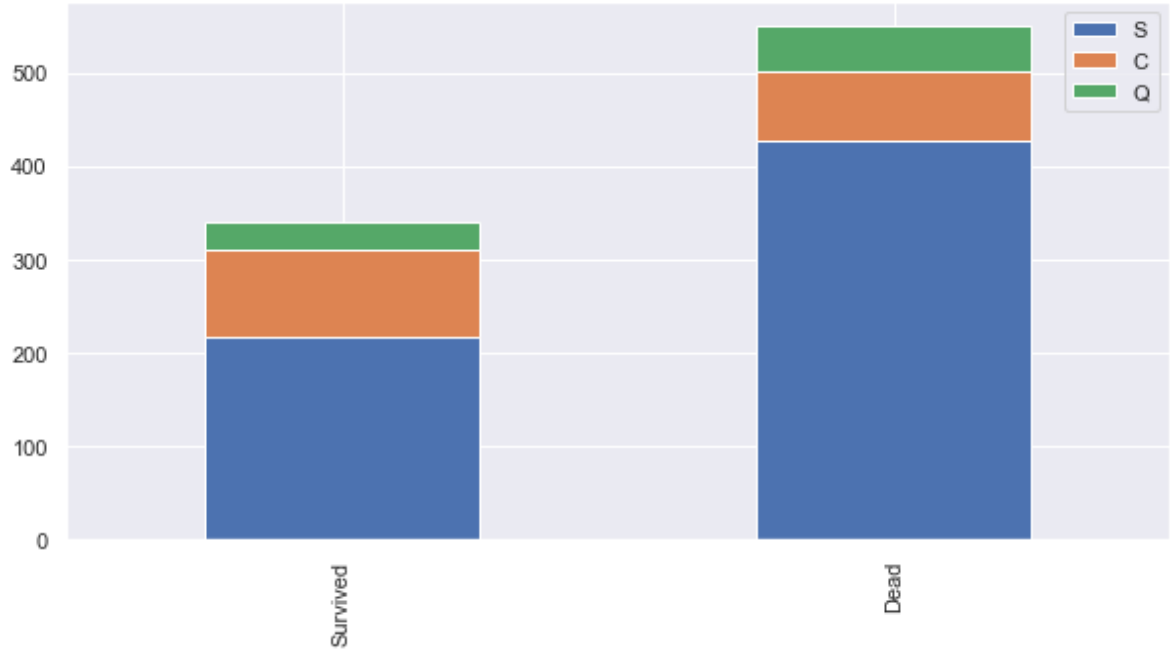
In [26]:

```
bar_chart('Parch')
```



In [27]:

```
bar_chart('Embarked')
```



In [31]:

```
train_df.head(10)
```

Out[31]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708

In [33]:

```
# Name

train_test_data = [train_df, test_df] # combining train and test dataset

for dataset in train_test_data:
    dataset['Title'] = dataset['Name'].str.extract(' ([A-Za-z]+)\.', expand=False)
```

In [34]:

```
train_df['Title'].value_counts()
```

Out[34]:

```
Mr          517
Miss        182
Mrs         125
Master       40
Dr            7
Rev           6
Col           2
Mlle          2
Major         2
Don           1
Lady          1
Countess      1
Mme           1
Ms            1
Sir           1
Capt          1
Jonkheer      1
Name: Title, dtype: int64
```

In [35]:

```
test_df['Title'].value_counts()
```

Out[35]:

```
Mr        240
Miss       78
Mrs        72
Master     21
Col         2
Rev         2
Dona        1
Ms          1
Dr          1
Name: Title, dtype: int64
```

In [36]:

```
title_mapping = {"Mr": 0, "Miss": 1, "Mrs": 2,
                 "Master": 3, "Dr": 3, "Rev": 3, "Col": 3, "Major": 3, "Mlle": 3,"Countess"
                 "Ms": 3, "Lady": 3, "Jonkheer": 3, "Don": 3, "Dona" : 3, "Mme": 3,"Capt":
for dataset in train_test_data:
    dataset['Title'] = dataset['Title'].map(title_mapping)
```

In [38]:

```
train_df.head()
```

Out[38]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	C
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	

In [39]:

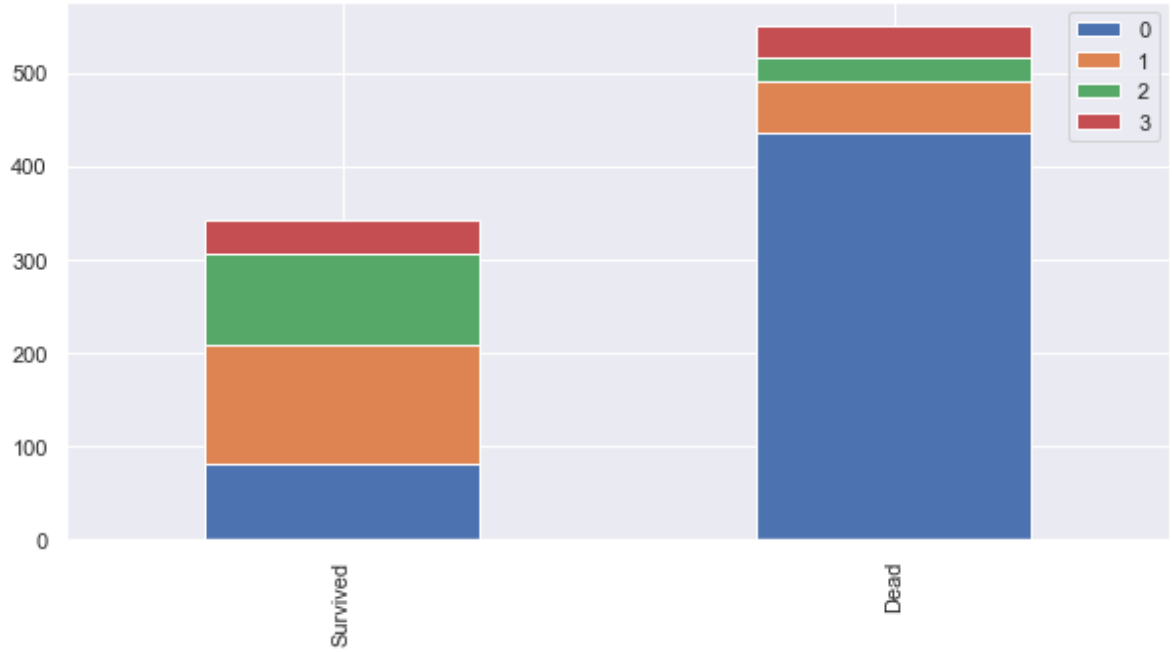
```
test_df.head()
```

Out[39]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Emba
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	

In [40]:

```
bar_chart('Title')
```



In [42]:

```
# delete unnecessary feature from dataset
train_df.drop('Name', axis=1, inplace=True)
test_df.drop('Name', axis=1, inplace=True)
```

In [44]:

```
train_df.head()
```

Out[44]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Emb
0	1	0	3	male	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	female	38.0	1	0	PC 17599	71.2833	C85	
2	3	1	3	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	4	1	1	female	35.0	1	0	113803	53.1000	C123	
4	5	0	3	male	35.0	0	0	373450	8.0500	NaN	

In [45]:

```
test_df.head()
```

Out[45]:

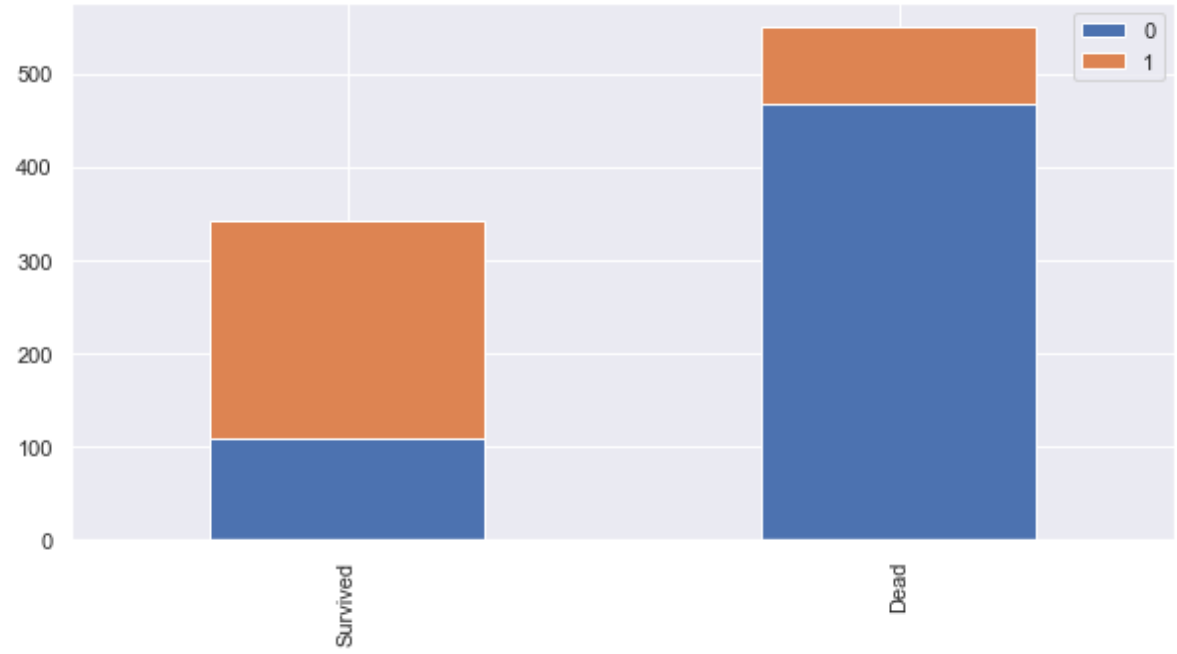
	PassengerId	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Title
0	892	3	male	34.5	0	0	330911	7.8292	NaN	Q	0
1	893	3	female	47.0	1	0	363272	7.0000	NaN	S	2
2	894	2	male	62.0	0	0	240276	9.6875	NaN	Q	0
3	895	3	male	27.0	0	0	315154	8.6625	NaN	S	0
4	896	3	female	22.0	1	1	3101298	12.2875	NaN	S	2

In [47]:

```
#Sex
sex_mapping = {"male": 0, "female": 1}
for dataset in train_test_data:
    dataset['Sex'] = dataset['Sex'].map(sex_mapping)
```

In [48]:

```
bar_chart('Sex')
```



In [50]:

```
#age
train_df.head(100)
```

Out[50]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Emba
0	1	0	3	0	22.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	1	38.0	1	0	PC 17599	71.2833	C85	
2	3	1	3	1	26.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	4	1	1	1	35.0	1	0	113803	53.1000	C123	
4	5	0	3	0	35.0	0	0	373450	8.0500	NaN	
...	...	...	...	...	...	...	...	...	...	...	
95	96	0	3	0	NaN	0	0	374910	8.0500	NaN	
96	97	0	1	0	71.0	0	0	PC 17754	34.6542	A5	
97	98	1	1	0	23.0	0	1	PC 17759	63.3583	D10 D12	
98	99	1	2	1	34.0	0	1	231919	23.0000	NaN	
99	100	0	2	0	34.0	1	0	244367	26.0000	NaN	

100 rows × 12 columns

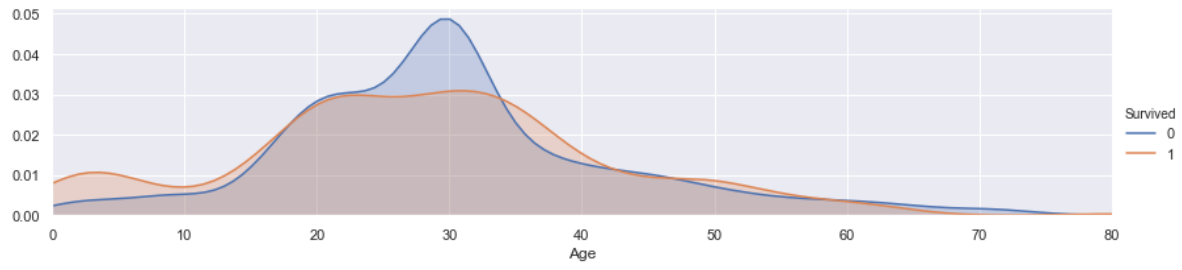
In [52]:

```
# fill missing age with median age for each title (Mr, Mrs, Miss, Others)
train_df["Age"].fillna(train_df.groupby("Title")["Age"].transform("median"), inplace=True)
test_df["Age"].fillna(test_df.groupby("Title")["Age"].transform("median"), inplace=True)
```

In [59]:

```
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()

plt.show()
```



In [63]:

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
PassengerId    891 non-null int64
Survived       891 non-null int64
Pclass         891 non-null int64
Sex            891 non-null int64
Age            891 non-null float64
SibSp          891 non-null int64
Parch          891 non-null int64
Ticket         891 non-null object
Fare           891 non-null float64
Cabin          204 non-null object
Embarked       889 non-null object
Title          891 non-null int64
dtypes: float64(2), int64(7), object(3)
memory usage: 83.7+ KB
```

In [64]:

```
test_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 11 columns):
PassengerId    418 non-null int64
Pclass         418 non-null int64
Sex            418 non-null int64
Age            418 non-null float64
SibSp          418 non-null int64
Parch          418 non-null int64
Ticket         418 non-null object
Fare           417 non-null float64
Cabin          91 non-null object
Embarked       418 non-null object
Title          418 non-null int64
dtypes: float64(2), int64(6), object(3)
memory usage: 36.0+ KB
```

In [65]:

```
for dataset in train_test_data:
    dataset.loc[ dataset['Age'] <= 16, 'Age'] = 0,
    dataset.loc[(dataset['Age'] > 16) & (dataset['Age'] <= 26), 'Age'] = 1,
    dataset.loc[(dataset['Age'] > 26) & (dataset['Age'] <= 36), 'Age'] = 2,
    dataset.loc[(dataset['Age'] > 36) & (dataset['Age'] <= 62), 'Age'] = 3,
    dataset.loc[ dataset['Age'] > 62, 'Age'] = 4
```

In [66]:

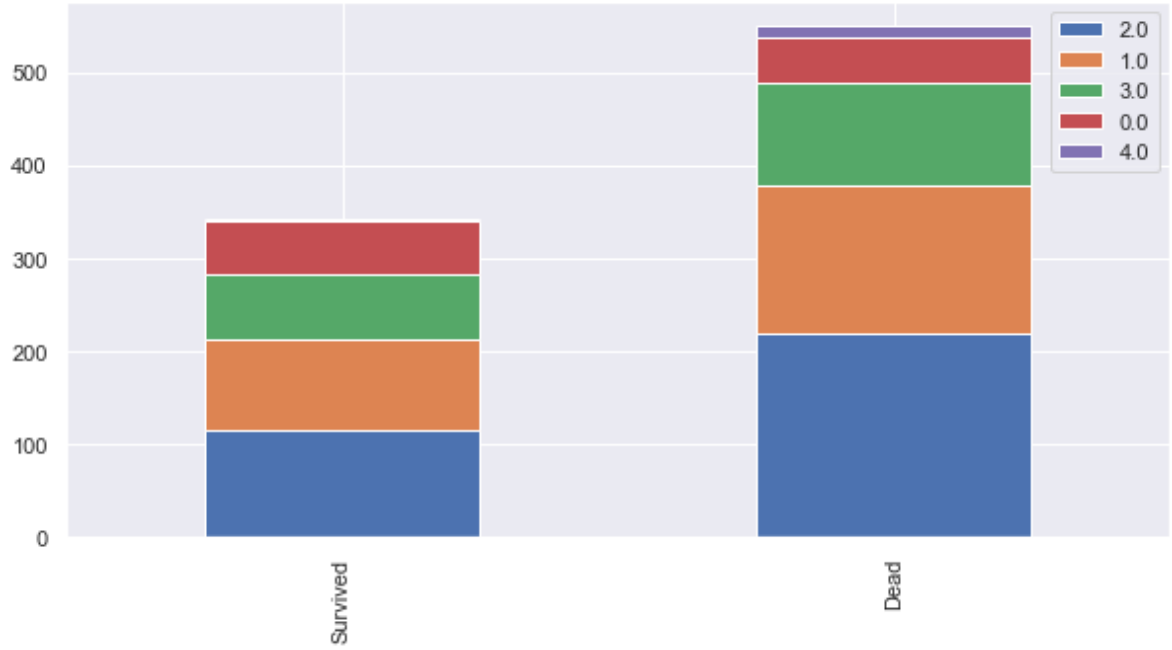
```
train_df.head()
```

Out[66]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embark
0	1	0	3	0	1.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	1	3.0	1	0	PC 17599	71.2833	C85	
2	3	1	3	1	1.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	4	1	1	1	2.0	1	0	113803	53.1000	C123	
4	5	0	3	0	2.0	0	0	373450	8.0500	NaN	

In [67]:

```
bar_chart('Age')
```

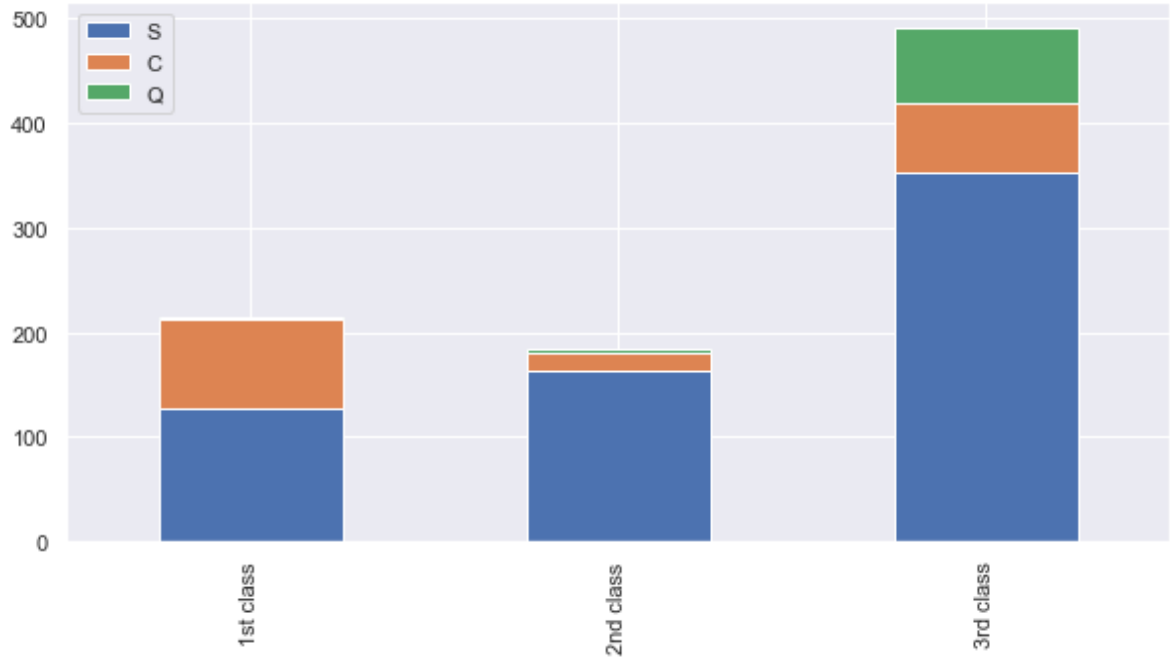


In [69]:

```
#Embarked
#filling missing values
Pclass1 = train_df[train_df['Pclass']==1]['Embarked'].value_counts()
Pclass2 = train_df[train_df['Pclass']==2]['Embarked'].value_counts()
Pclass3 = train_df[train_df['Pclass']==3]['Embarked'].value_counts()
df = pd.DataFrame([Pclass1, Pclass2, Pclass3])
df.index = ['1st class','2nd class', '3rd class']
df.plot(kind='bar',stacked=True, figsize=(10,5))
```

Out[69]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xd3dc908>
```



In [73]:

```
#fill out missing embark with S embark
for dataset in train_test_data:
    dataset['Embarked'] = dataset['Embarked'].fillna('S')
```

In [72]:

```
train_df.head()
```

Out[72]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embark
0	1	0	3	0	1.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	1	3.0	1	0	PC 17599	71.2833	C85	
2	3	1	3	1	1.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	4	1	1	1	2.0	1	0	113803	53.1000	C123	
4	5	0	3	0	2.0	0	0	373450	8.0500	NaN	

In [74]:

```
embarked_mapping = {"S": 0, "C": 1, "Q": 2}
for dataset in train_test_data:
    dataset['Embarked'] = dataset['Embarked'].map(embarked_mapping)
```

In [76]:

```
#fare
# fill missing Fare with median fare for each Pclass
train_df["Fare"].fillna(train_df.groupby("Pclass")["Fare"].transform("median"), inplace=Tru
test_df["Fare"].fillna(test_df.groupby("Pclass")["Fare"].transform("median"), inplace=True)
train_df.head(50)
```

Out[76]:

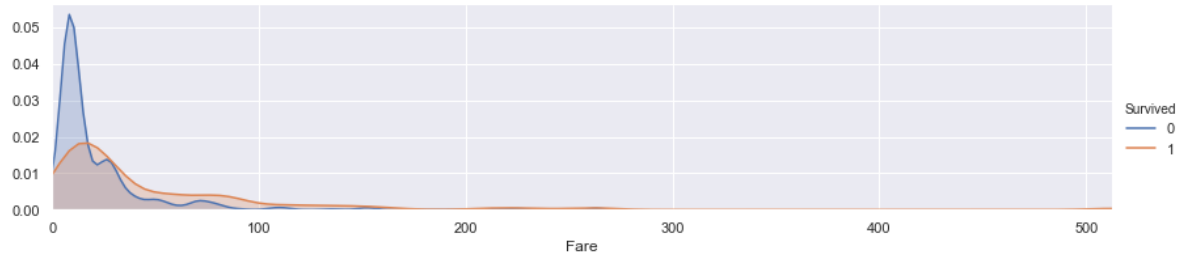
	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Emb
0	1	0	3	0	1.0	1	0	A/5 21171	7.2500	NaN	
1	2	1	1	1	3.0	1	0	PC 17599	71.2833	C85	
2	3	1	3	1	1.0	0	0	STON/O2. 3101282	7.9250	NaN	
3	4	1	1	1	2.0	1	0	113803	53.1000	C123	
4	5	0	3	0	2.0	0	0	373450	8.0500	NaN	
5	6	0	3	0	2.0	0	0	330877	8.4583	NaN	
6	7	0	1	0	3.0	0	0	17463	51.8625	E46	
7	8	0	3	0	0.0	3	1	349909	21.0750	NaN	
8	9	1	3	1	2.0	0	2	347742	11.1333	NaN	
9	10	1	2	1	0.0	1	0	237736	30.0708	NaN	
10	11	1	3	1	0.0	1	1	PP 9549	16.7000	G6	
11	12	1	1	1	3.0	0	0	113783	26.5500	C103	
12	13	0	3	0	1.0	0	0	A/5. 2151	8.0500	NaN	
13	14	0	3	0	3.0	1	5	347082	31.2750	NaN	
14	15	0	3	1	0.0	0	0	350406	7.8542	NaN	
15	16	1	2	1	3.0	0	0	248706	16.0000	NaN	
16	17	0	3	0	0.0	4	1	382652	29.1250	NaN	
17	18	1	2	0	2.0	0	0	244373	13.0000	NaN	
18	19	0	3	1	2.0	1	0	345763	18.0000	NaN	
19	20	1	3	1	2.0	0	0	2649	7.2250	NaN	
20	21	0	2	0	2.0	0	0	239865	26.0000	NaN	
21	22	1	2	0	2.0	0	0	248698	13.0000	D56	
22	23	1	3	1	0.0	0	0	330923	8.0292	NaN	
23	24	1	1	0	2.0	0	0	113788	35.5000	A6	
24	25	0	3	1	0.0	3	1	349909	21.0750	NaN	
25	26	1	3	1	3.0	1	5	347077	31.3875	NaN	
26	27	0	3	0	2.0	0	0	2631	7.2250	NaN	
27	28	0	1	0	1.0	3	2	19950	263.0000	C23 C25 C27	
28	29	1	3	1	1.0	0	0	330959	7.8792	NaN	
29	30	0	3	0	2.0	0	0	349216	7.8958	NaN	

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Emba
30	31	0	1	0	3.0	0	0	PC 17601	27.7208	NaN	
31	32	1	1	1	2.0	1	0	PC 17569	146.5208	B78	
32	33	1	3	1	1.0	0	0	335677	7.7500	NaN	
33	34	0	2	0	4.0	0	0	C.A. 24579	10.5000	NaN	
34	35	0	1	0	2.0	1	0	PC 17604	82.1708	NaN	
35	36	0	1	0	3.0	1	0	113789	52.0000	NaN	
36	37	1	3	0	2.0	0	0	2677	7.2292	NaN	
37	38	0	3	0	1.0	0	0	A./5. 2152	8.0500	NaN	
38	39	0	3	1	1.0	2	0	345764	18.0000	NaN	
39	40	1	3	1	0.0	1	0	2651	11.2417	NaN	
40	41	0	3	1	3.0	1	0	7546	9.4750	NaN	
41	42	0	2	1	2.0	1	0	11668	21.0000	NaN	
42	43	0	3	0	2.0	0	0	349253	7.8958	NaN	
43	44	1	2	1	0.0	1	2	SC/Paris 2123	41.5792	NaN	
44	45	1	3	1	1.0	0	0	330958	7.8792	NaN	
45	46	0	3	0	2.0	0	0	S.C./A.4. 23567	8.0500	NaN	
46	47	0	3	0	2.0	1	0	370371	15.5000	NaN	
47	48	1	3	1	1.0	0	0	14311	7.7500	NaN	
48	49	0	3	0	2.0	2	0	2662	21.6792	NaN	
49	50	0	3	1	1.0	1	0	349237	17.8000	NaN	

In [80]:

```
facet = sns.FacetGrid(train_df, hue="Survived",aspect=4)
facet.map(sns.kdeplot,'Fare',shade= True)
facet.set(xlim=(0, train_df['Fare'].max()))
facet.add_legend()

plt.show()
```

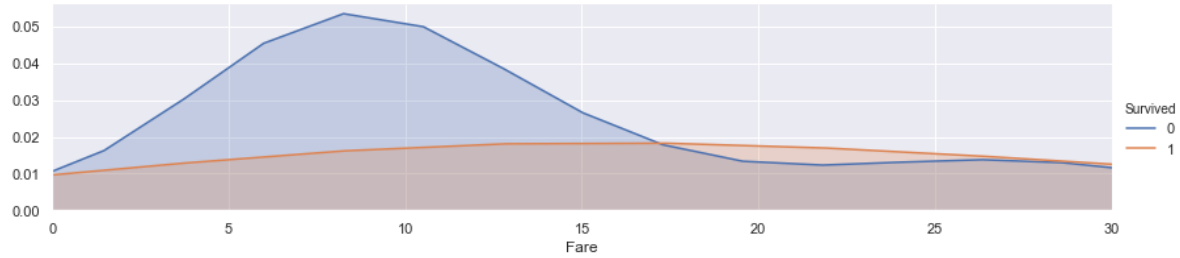


In [87]:

```
facet = sns.FacetGrid(train_df, hue="Survived",aspect=4)
facet.map(sns.kdeplot,'Fare',shade= True)
facet.set(xlim=(0, train_df['Fare'].max()))
facet.add_legend()
plt.xlim(0, 30)
```

Out[87]:

```
(0, 30)
```



In [88]:

```
for dataset in train_test_data:
    dataset.loc[ dataset['Fare'] <= 17, 'Fare'] = 0,
    dataset.loc[(dataset['Fare'] > 17) & (dataset['Fare'] <= 30), 'Fare'] = 1,
    dataset.loc[(dataset['Fare'] > 30) & (dataset['Fare'] <= 100), 'Fare'] = 2,
    dataset.loc[ dataset['Fare'] > 100, 'Fare'] = 3
```

In [89]:

```
train_df.head()
```

Out[89]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	0	1.0	1	0	A/5 21171	0.0	NaN	0
1	2	1	1	1	3.0	1	0	PC 17599	2.0	C85	1
2	3	1	3	1	1.0	0	0	STON/O2. 3101282	0.0	NaN	0
3	4	1	1	1	2.0	1	0	113803	2.0	C123	0
4	5	0	3	0	2.0	0	0	373450	0.0	NaN	0

In [90]:

```
#cabin
train_df.Cabin.value_counts()
```

Out[90]:

```
B96 B98        4
C23 C25 C27    4
G6             4
C22 C26        3
D              3
              ..
E34            1
B80            1
B78            1
B73            1
D49            1
Name: Cabin, Length: 147, dtype: int64
```

In [91]:

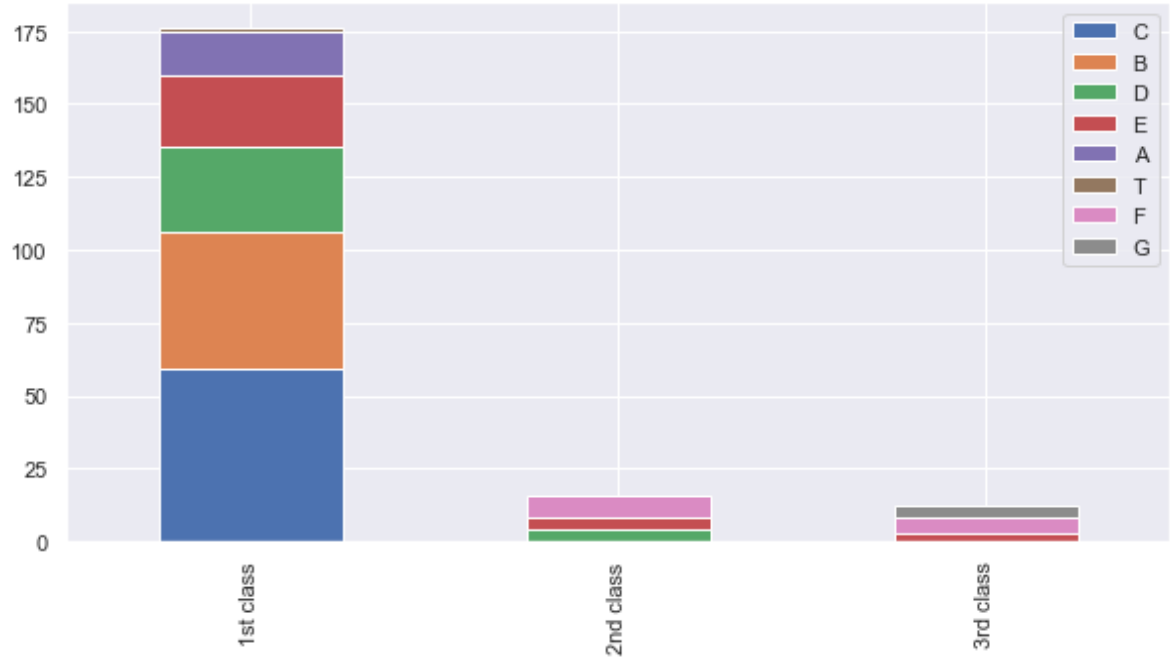
```
for dataset in train_test_data:
    dataset['Cabin'] = dataset['Cabin'].str[:1]
```

In [93]:

```
Pclass1 = train_df[train_df['Pclass']==1]['Cabin'].value_counts()
Pclass2 = train_df[train_df['Pclass']==2]['Cabin'].value_counts()
Pclass3 = train_df[train_df['Pclass']==3]['Cabin'].value_counts()
df = pd.DataFrame([Pclass1, Pclass2, Pclass3])
df.index = ['1st class','2nd class', '3rd class']
df.plot(kind='bar',stacked=True, figsize=(10,5))
```

Out[93]:

```
<matplotlib.axes._subplots.AxesSubplot at 0xdf7e708>
```



In [94]:

```
cabin_mapping = {"A": 0, "B": 0.4, "C": 0.8, "D": 1.2, "E": 1.6, "F": 2, "G": 2.4, "T": 2.8
for dataset in train_test_data:
    dataset['Cabin'] = dataset['Cabin'].map(cabin_mapping)
```

In [96]:

```
# fill missing Fare with median fare for each Pclass
train_df["Cabin"].fillna(train_df.groupby("Pclass")["Cabin"].transform("median"), inplace=T
test_df["Cabin"].fillna(test_df.groupby("Pclass")["Cabin"].transform("median"), inplace=Tru
```

In [98]:

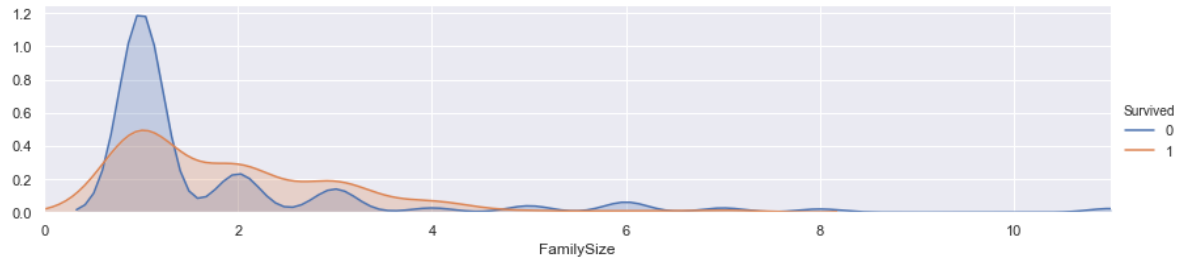
```
#family size
train_df["FamilySize"] = train_df["SibSp"] + train_df["Parch"] + 1
test_df["FamilySize"] = test_df["SibSp"] + test_df["Parch"] + 1
```

In [99]:

```
facet = sns.FacetGrid(train_df, hue="Survived",aspect=4)
facet.map(sns.kdeplot,'FamilySize',shade= True)
facet.set(xlim=(0, train_df['FamilySize'].max()))
facet.add_legend()
plt.xlim(0)
```

Out[99]:

```
(0, 11.0)
```



In [100]:

```
family_mapping = {1: 0, 2: 0.4, 3: 0.8, 4: 1.2, 5: 1.6, 6: 2, 7: 2.4, 8: 2.8, 9: 3.2, 10: 3
for dataset in train_test_data:
    dataset['FamilySize'] = dataset['FamilySize'].map(family_mapping)
```

In [101]:

```
train_df.head()
```

Out[101]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	0	1.0	1	0	A/5 21171	0.0	2.0	0
1	2	1	1	1	3.0	1	0	PC 17599	2.0	0.8	1
2	3	1	3	1	1.0	0	0	STON/O2. 3101282	0.0	2.0	0
3	4	1	1	1	2.0	1	0	113803	2.0	0.8	0
4	5	0	3	0	2.0	0	0	373450	0.0	2.0	0

In [103]:

```
features_drop = ['Ticket', 'SibSp', 'Parch']
train_df = train_df.drop(features_drop, axis=1)
test_df = test_df.drop(features_drop, axis=1)
train_df = train_df.drop(['PassengerId'], axis=1)
```

In [104]:

```
train_data = train_df.drop('Survived', axis=1)
target = train_df['Survived']

train_data.shape, target.shape
```

Out[104]:

```
((891, 8), (891,))
```

In [105]:

```
train_data.head(10)
```

Out[105]:

	Pclass	Sex	Age	Fare	Cabin	Embarked	Title	FamilySize
0	3	0	1.0	0.0	2.0	0	0	0.4
1	1	1	3.0	2.0	0.8	1	2	0.4
2	3	1	1.0	0.0	2.0	0	1	0.0
3	1	1	2.0	2.0	0.8	0	2	0.4
4	3	0	2.0	0.0	2.0	0	0	0.0
5	3	0	2.0	0.0	2.0	2	0	0.0
6	1	0	3.0	2.0	1.6	0	0	0.0
7	3	0	0.0	1.0	2.0	0	3	1.6
8	3	1	2.0	0.0	2.0	0	2	0.8
9	2	1	0.0	2.0	1.8	1	2	0.4

In [106]:

```
# Modelling
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
```

In [107]:

```
train_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 9 columns):
Survived      891 non-null int64
Pclass        891 non-null int64
Sex           891 non-null int64
Age           891 non-null float64
Fare          891 non-null float64
Cabin         891 non-null float64
Embarked      891 non-null int64
Title         891 non-null int64
FamilySize    891 non-null float64
dtypes: float64(4), int64(5)
memory usage: 62.8 KB
```

In [128]:

```
#Cross Validation (K-fold)
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
k_fold = KFold(n_splits=10, shuffle=True, random_state=0)
```

In [109]:

```
#kNN
clf = KNeighborsClassifier(n_neighbors = 13)
scoring = 'accuracy'
score = cross_val_score(clf, train_data, target, cv=k_fold, n_jobs=1, scoring=scoring)
print(score)
```

```
[0.82222222 0.76404494 0.80898876 0.83146067 0.87640449 0.82022472
 0.85393258 0.79775281 0.84269663 0.84269663]
```

In [110]:

```
# kNN Score
round(np.mean(score)*100, 2)
```

Out[110]:

```
82.6
```

In [111]:

```
#Decision Tree
clf = DecisionTreeClassifier()
scoring = 'accuracy'
score = cross_val_score(clf, train_data, target, cv=k_fold, n_jobs=1, scoring=scoring)
print(score)
```

```
[0.76666667 0.83146067 0.7752809  0.7752809  0.87640449 0.76404494
 0.83146067 0.82022472 0.74157303 0.78651685]
```

In [112]:

```
# decision tree Score
round(np.mean(score)*100, 2)
```

Out[112]:

```
79.69
```

In [113]:

```
#Ramdom Forest
clf = RandomForestClassifier(n_estimators=13)
scoring = 'accuracy'
score = cross_val_score(clf, train_data, target, cv=k_fold, n_jobs=1, scoring=scoring)
print(score)
```

```
[0.78888889 0.79775281 0.82022472 0.76404494 0.86516854 0.82022472
 0.82022472 0.79775281 0.7752809  0.83146067]
```

In [114]:

```
# Random Forest Score
round(np.mean(score)*100, 2)
```

Out[114]:

```
80.81
```

In [115]:

```
#Naive Bayes
clf = GaussianNB()
scoring = 'accuracy'
score = cross_val_score(clf, train_data, target, cv=k_fold, n_jobs=1, scoring=scoring)
print(score)
```

```
[0.85555556 0.73033708 0.75280899 0.75280899 0.70786517 0.80898876
 0.76404494 0.80898876 0.86516854 0.83146067]
```

In [116]:

```
## Naive Bayes Score
round(np.mean(score)*100, 2)
```

Out[116]:

```
78.78
```

In [117]:

```
#SVM
clf = SVC()
scoring = 'accuracy'
score = cross_val_score(clf, train_data, target, cv=k_fold, n_jobs=1, scoring=scoring)
print(score)
```

```
E:\Ana\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The default
value of gamma will change from 'auto' to 'scale' in version 0.22 to account
better for unscaled features. Set gamma explicitly to 'auto' or 'scale' to a
void this warning.
  "avoid this warning.", FutureWarning)
E:\Ana\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The default
value of gamma will change from 'auto' to 'scale' in version 0.22 to account
better for unscaled features. Set gamma explicitly to 'auto' or 'scale' to a
void this warning.
  "avoid this warning.", FutureWarning)
E:\Ana\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The default
value of gamma will change from 'auto' to 'scale' in version 0.22 to account
better for unscaled features. Set gamma explicitly to 'auto' or 'scale' to a
void this warning.
  "avoid this warning.", FutureWarning)
E:\Ana\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The default
value of gamma will change from 'auto' to 'scale' in version 0.22 to account
better for unscaled features. Set gamma explicitly to 'auto' or 'scale' to a
void this warning.
  "avoid this warning.", FutureWarning)
E:\Ana\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The default
value of gamma will change from 'auto' to 'scale' in version 0.22 to account
better for unscaled features. Set gamma explicitly to 'auto' or 'scale' to a
void this warning.
  "avoid this warning.", FutureWarning)

[0.83333333 0.80898876 0.83146067 0.82022472 0.84269663 0.82022472
 0.84269663 0.85393258 0.83146067 0.86516854]

E:\Ana\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The defau
lt value of gamma will change from 'auto' to 'scale' in version 0.22 to ac
count better for unscaled features. Set gamma explicitly to 'auto' or 'sca
le' to avoid this warning.
  "avoid this warning.", FutureWarning)
E:\Ana\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The defau
lt value of gamma will change from 'auto' to 'scale' in version 0.22 to ac
count better for unscaled features. Set gamma explicitly to 'auto' or 'sca
le' to avoid this warning.
  "avoid this warning.", FutureWarning)
E:\Ana\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The defau
lt value of gamma will change from 'auto' to 'scale' in version 0.22 to ac
count better for unscaled features. Set gamma explicitly to 'auto' or 'sca
le' to avoid this warning.
  "avoid this warning.", FutureWarning)
E:\Ana\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The defau
lt value of gamma will change from 'auto' to 'scale' in version 0.22 to ac
count better for unscaled features. Set gamma explicitly to 'auto' or 'sca
le' to avoid this warning.
  "avoid this warning.", FutureWarning)
E:\Ana\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The defau
lt value of gamma will change from 'auto' to 'scale' in version 0.22 to ac
count better for unscaled features. Set gamma explicitly to 'auto' or 'sca
```

```
le' to avoid this warning.
```

In [118]:

```
round(np.mean(score)*100,2)
```

Out[118]:

```
83.5
```

In [ ]:

```
#SVM=83.5 highest
```

In [130]:

```
#Testing
clf = SVC()
clf.fit(train_data, target)

test_data = test_df.drop("PassengerId", axis=1).copy()
prediction = clf.predict(test_data)
```

```
E:\Ana\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The default
value of gamma will change from 'auto' to 'scale' in version 0.22 to account
better for unscaled features. Set gamma explicitly to 'auto' or 'scale' to a
void this warning.
  "avoid this warning.", FutureWarning)
```

In [126]:

```
submission = pd.DataFrame({
        "PassengerId": test_df["PassengerId"],
        "Survived": prediction
    })

submission.to_csv('submission.csv', index=False)
```

In [127]:

```
submission = pd.read_csv('submission.csv')
submission.head()
```

Out[127]:

	PassengerId	Survived
0	892	0
1	893	1
2	894	0
3	895	0
4	896	1

In [ ]:

```

```