ANALSYIS OF TITANIC DATASET

In [7]:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns
```

Read 'Training' and 'Test' data files

In [8]:

```
train_data = pd.read_csv("../input/titanic/train.csv")
test_data = pd.read_csv("../input/titanic/test.csv")

combine = [train_data, test_data]
```

How data looks

In [9]:

```
train_data.head()
```

Out[9]:

	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

Some information about data

In [10]:

```
print(train_data.info())

print('\nNumber of rows : ', train_data.shape[0])
print('Number of columns : ', train_data.shape[1])

print('\nTrain columns with null values:')
train_data.isnull().sum()

print('\nNumber of total people who survived or dead ( 0 : Dead, 1: Survived )')
train_data['Survived'].value_counts().apply(lambda x:f'{x} ({x*100/len(train_data):0.2f}
%)')

women = train_data.loc[train_data.Sex == 'female']["Survived"]
rate_women = sum(women)/len(women)
print("\n% of women who survived:", rate_women)

men = train_data.loc[train_data.Sex == 'male']["Survived"]
rate_men = sum(men)/len(men)
```

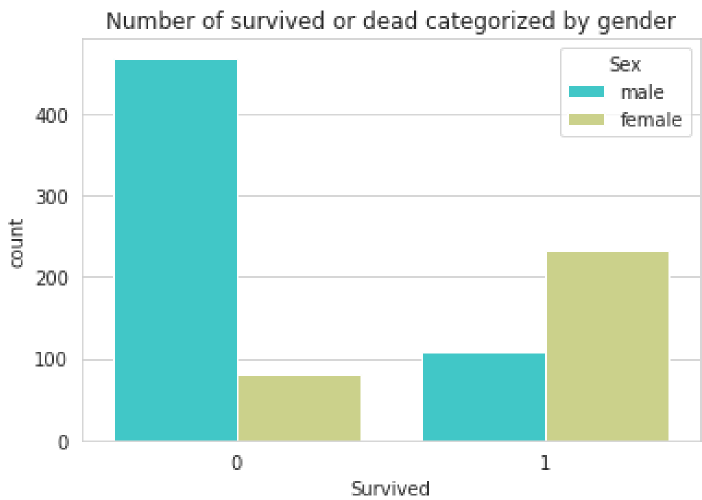
```
print("\n% of men who survived:", rate_men)
```

```
<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
None

Number of rows :  891
Number of columns :  12

Train columns with null values:

Number of total people who survived or dead ( 0 : Dead, 1: Survived )

% of women who survived: 0.7420382165605095

% of men who survived: 0.18890814558058924
```

DATA VISUALIZATION

In [11]:

```
sns.set_style('whitegrid')
s = sns.countplot(x='Survived', hue='Sex', data=train_data, palette='rainbow')
s.set_title("Number of survived or dead categorized by gender")
plt.show()
print("> We see that survival rate of women is more than men")
```



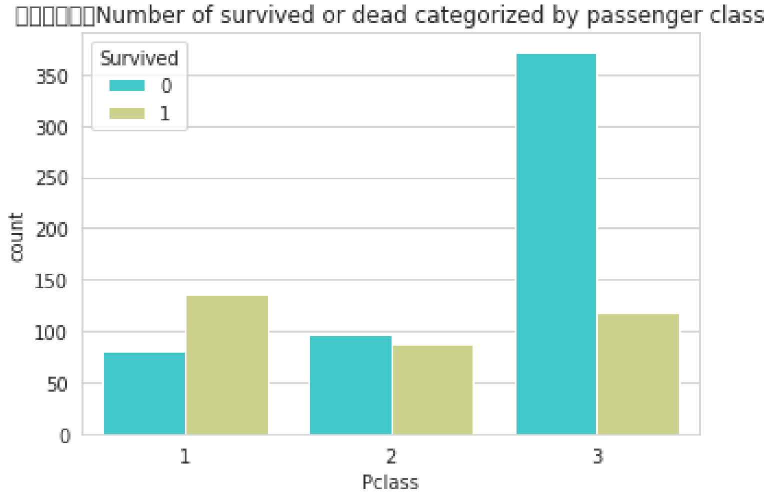

```
> We see that survival rate of women is more than men
```

In [12]:

```
sns.set_style('whitegrid')
s = sns.countplot(x='Pclass', hue='Survived', data=train_data, palette='rainbow')
s.set_title("\t\t\t\t\t\t\tNumber of survived or dead categorized by passenger class")
plt.show()
```

```
print("> We see that survival rate of 1st and 2nd passenger class is higher than 3rd clas
s")
```

```
/opt/conda/lib/python3.7/site-packages/matplotlib/backends/backend_agg.py:240: RuntimeWar
ning: Glyph 9 missing from current font.
  font.set_text(s, 0.0, flags=flags)
/opt/conda/lib/python3.7/site-packages/matplotlib/backends/backend_agg.py:203: RuntimeWar
ning: Glyph 9 missing from current font.
  font.set_text(s, 0, flags=flags)
```

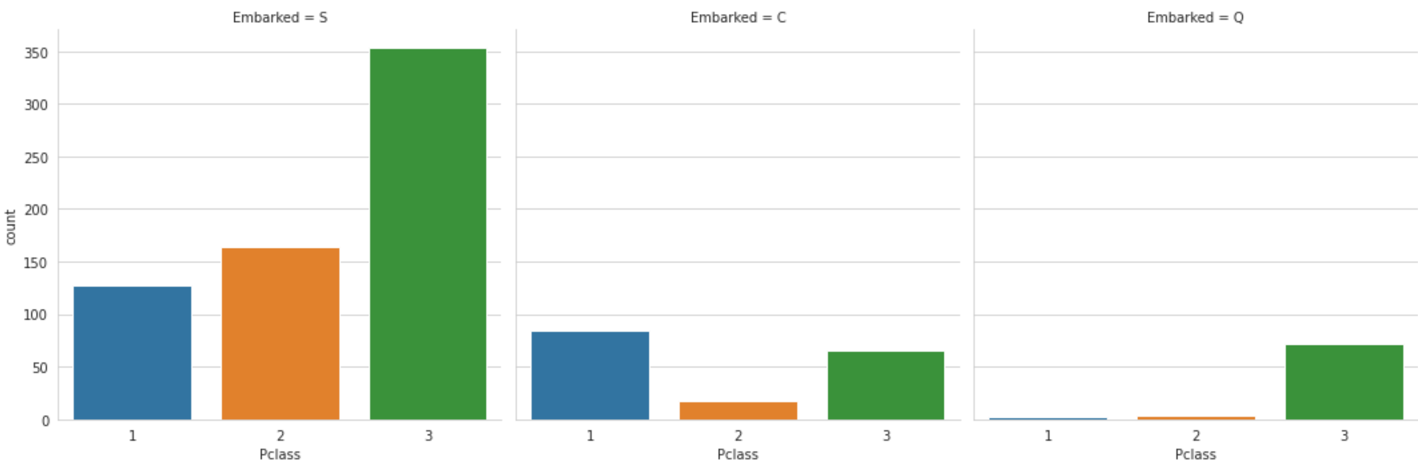



```
> We see that survival rate of 1st and 2nd passenger class is higher than 3rd class
```

In [13]:

```
sns.set_style('whitegrid')
sns.catplot(x='Pclass', col='Embarked', kind='count', data=train_data)
print("Number of people categorized by port of embarkation")
plt.show()
print("Hint : C = Cherbourg, Q = Queenstown, S = Southampton ")
print("> We see that most of people come from Southampton and chose 3rd class")
```

```
Number of people categorized by port of embarkation
```



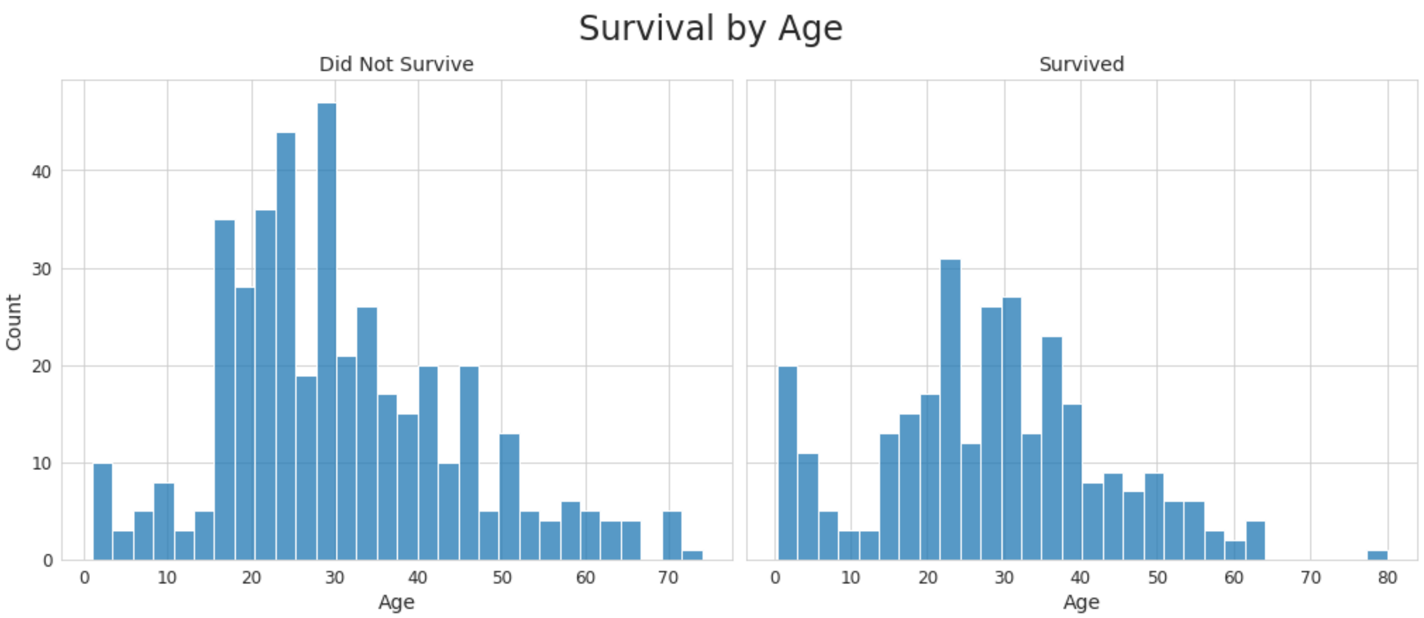

```
Hint : C = Cherbourg, Q = Queenstown, S = Southampton
> We see that most of people come from Southampton and chose 3rd class
```

In [14]:

```
fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(14,6), constrained_layout=True, shar
ey=True)

#Graphs
age_surv = sns.histplot(data=train_data[train_data['Survived']==1],x='Age',bins=30,ax=ax
es[1])
age_dead = sns.histplot(data=train_data[train_data['Survived']==0],x='Age',bins=30,ax=ax
es[0])

figtitle = fig.suptitle('Survival by Age',fontsize=24)
axeszero_ylabel = axes[0].set_ylabel('Count',size=14)
```

```
axeszero_yticklabel = axes[0].set_yticklabels([int(x) for x in age_dead.get_yticks()],size=12)
axeszero_title = axes[0].set_title('Did Not Survive', fontsize=14)
axeszero_xticklabel = axes[0].set_xticklabels([int(x) for x in age_dead.get_xticks()],size=12)
axeszero_xlabel = axes[0].set_xlabel('Age',size=14)
axesone_title = axes[1].set_title('Survived', fontsize=14)
axesone_xticklabel = axes[1].set_xticklabels([int(x) for x in age_surv.get_xticks()],size=12)
axesone_xlabel = axes[1].set_xlabel('Age',size=14)
plt.show()
print("> We see that between 20-30 ages almost same probability of survived and didn't survived")
```

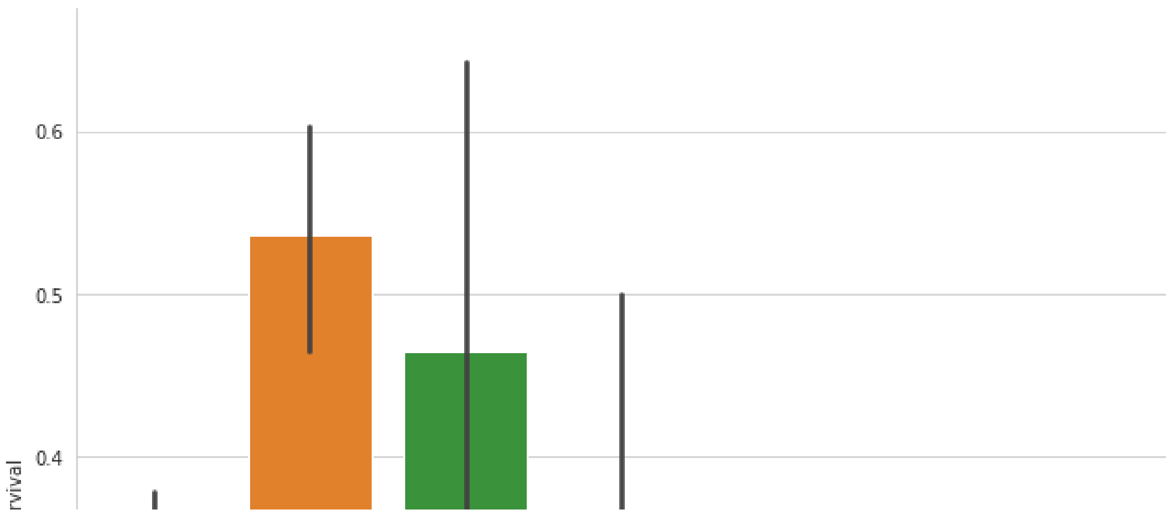
```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:9: UserWarning: FixedFormatter should only be used together with FixedLocator
  if __name__ == '__main__':
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:11: UserWarning: FixedFormatter should only be used together with FixedLocator
  # This is added back by InteractiveShellApp.init_path()
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:14: UserWarning: FixedFormatter should only be used together with FixedLocator
```

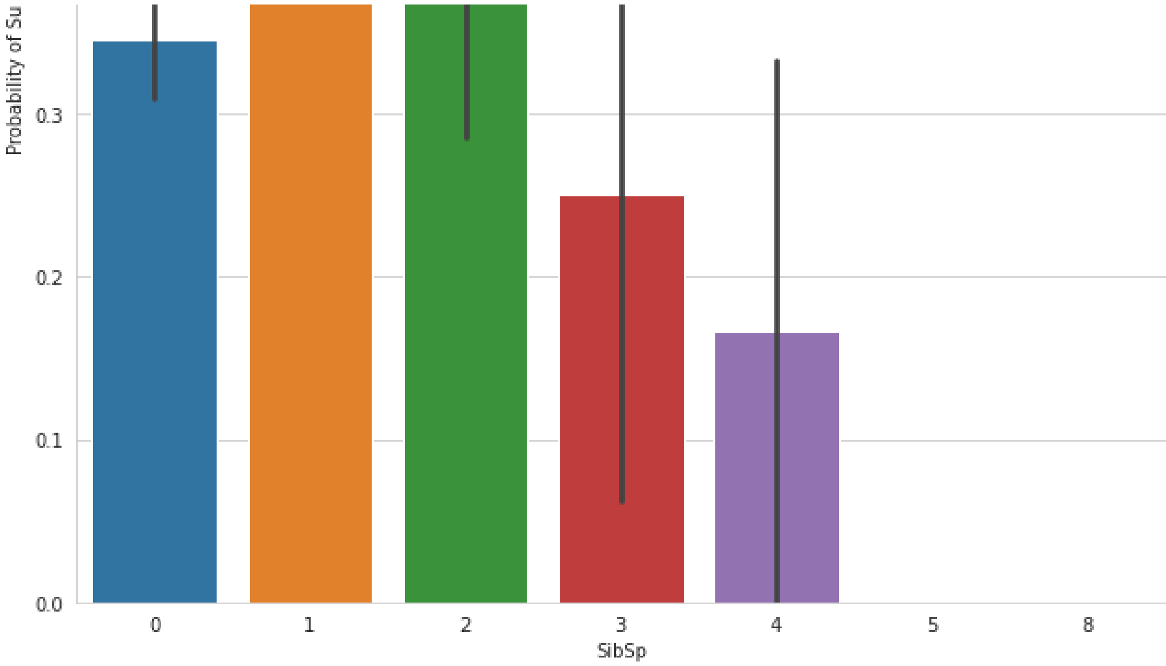



```
> We see that between 20-30 ages almost same probability of survived and didn't survived
```

In [15]:

```
#SibSp - Survived
sns.set_style('whitegrid')
g = sns.catplot(x = "SibSp", y = "Survived", data = train_data, kind = "bar", height= 9)
g.set_ylabels("Probability of Survival")
plt.show()
print("Hint : SibSp = number of siblings / spouses ")
```

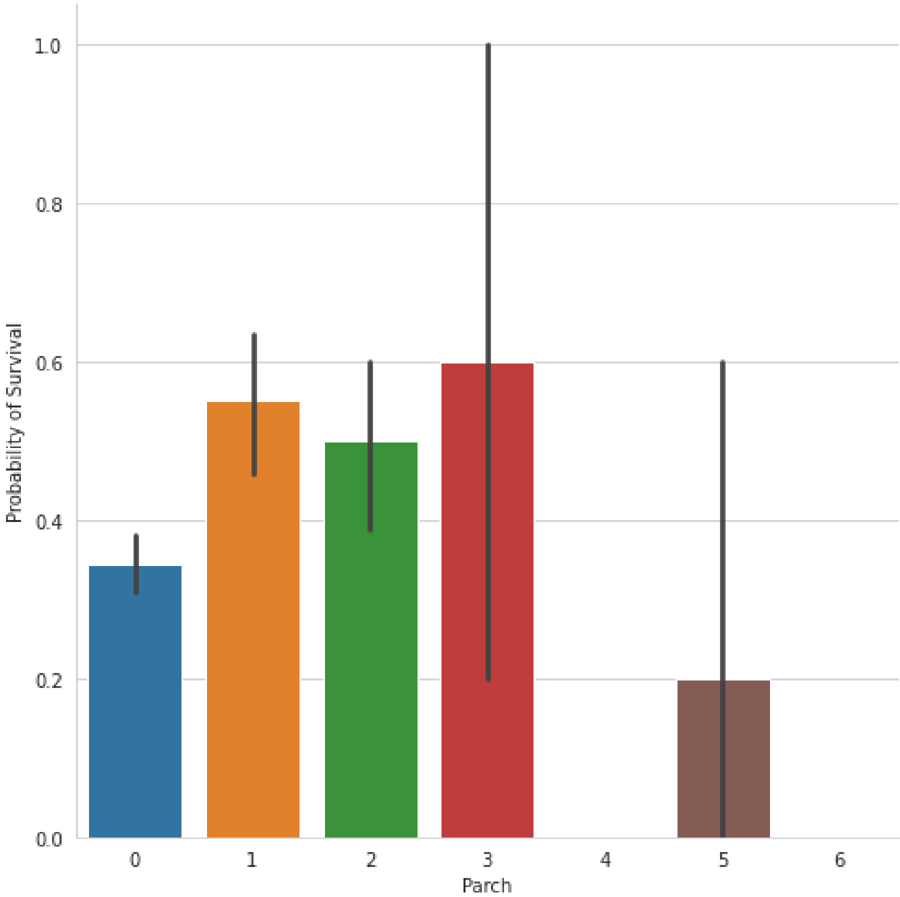





```
Hint : SibSp = number of siblings / spouses
```

In [16]:

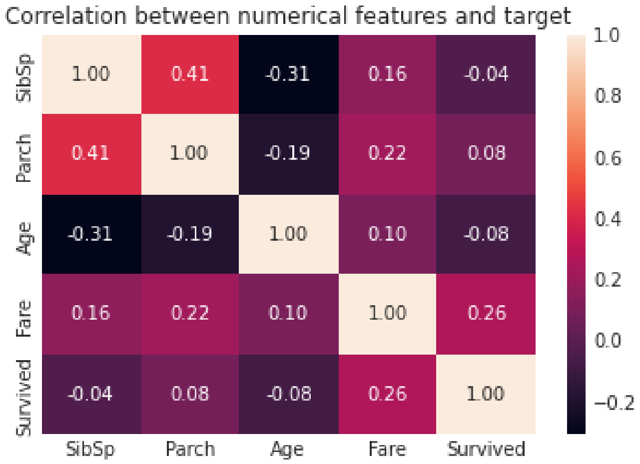
```
#ParCh - Survived
sns.set_style('whitegrid')
g = sns.catplot(x = "Parch", y = "Survived", data = train_data, kind = "bar", height = 7
)
g.set_ylabels("Probability of Survival")
plt.show()
print("Hint : Parch = number of parents / children ")
print("> After we see that from last two visualization, small families have more chance t
o survive")
```




```
Hint : Parch = number of parents / children
> After we see that from last two visualization, small families have more chance to survi
ve
```

In [17]:

```
plt.style.use("seaborn-whitegrid")
num_col = ["SibSp", "Parch", "Age", "Fare", "Survived"]
s = sns.heatmap(train_data[num_col].corr(), annot = True, fmt = ".2f")
s.set_title("Correlation between numerical features and target")
plt.show()
print("> We see that these features have not more affect to target")
```




```
> We see that these features have not more affect to target
```

FEATURE ENGINEERING

In [18]:

```
print('\nTrain data with null values:')
train_data.isnull().sum()

print('\nTest data with null values:')
test_data.isnull().sum()
```

```
Train data with null values:

Test data with null values:
```

Out[18]:

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

NAME - TITLE SPLIT and MAPPING

In [19]:

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

# SITUATION OF TITLE COLUMN AND NUMBER OF EACH TITLE
train_data['Title'].value_counts()
```

Out[19]:

```
Mr          517
Miss        182
Mrs         125
```

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

> **Depends on these values mapping for title column**
>
> - **Mr : 0**
> - **Miss : 1**
> - **Mrs: 2**
> - **Master: 3**
> - **Others : 4**

In [20]:

```
title_mapping = {"Mr": 0, "Miss": 1, "Mrs": 2,
                 "Master": 3, "Dr": 4, "Rev": 4, "Col": 4, "Major": 4, "Mlle": 4,"Counte
ss": 4,
                 "Ms": 4, "Lady": 4, "Jonkheer": 4, "Don": 4, "Dona" : 4, "Mme": 4,"Capt
": 4,"Sir": 4 }
for dataset in combine:
    dataset['Title'] = dataset['Title'].map(title_mapping)
```

FAMILY SIZE

In [21]:

```
train_data['family_size'] = train_data['SibSp'] + train_data['Parch'] + 1
test_data['family_size'] = test_data['SibSp'] + test_data['Parch'] + 1
```

> **It's calculated by addition of SibSp and Parch number (+ 1 means that family at least one member)**

GENDER MAPPING

In [22]:

```
for dataset in combine:
    dataset['Sex'] = dataset['Sex'].map( {'female': 1, 'male': 0} ).astype(int)
```

> **0 assigned to female and 1 assigned to the male person**

HAS CABIN

In [23]:

```
#CABIN
```

```
for dataset in combine:
    dataset['Has_Cabin'] = dataset["Cabin"].apply(lambda x: 0 if type(x) == float else 1
)
```

> **Cabin column has too much missing values. So mapping for has cabin or not seems better solution. Because prediction of these missing values looks impossible and if we try to fill up them, it can directly affect learning of model as positively or negatively. So it's unguessable.**

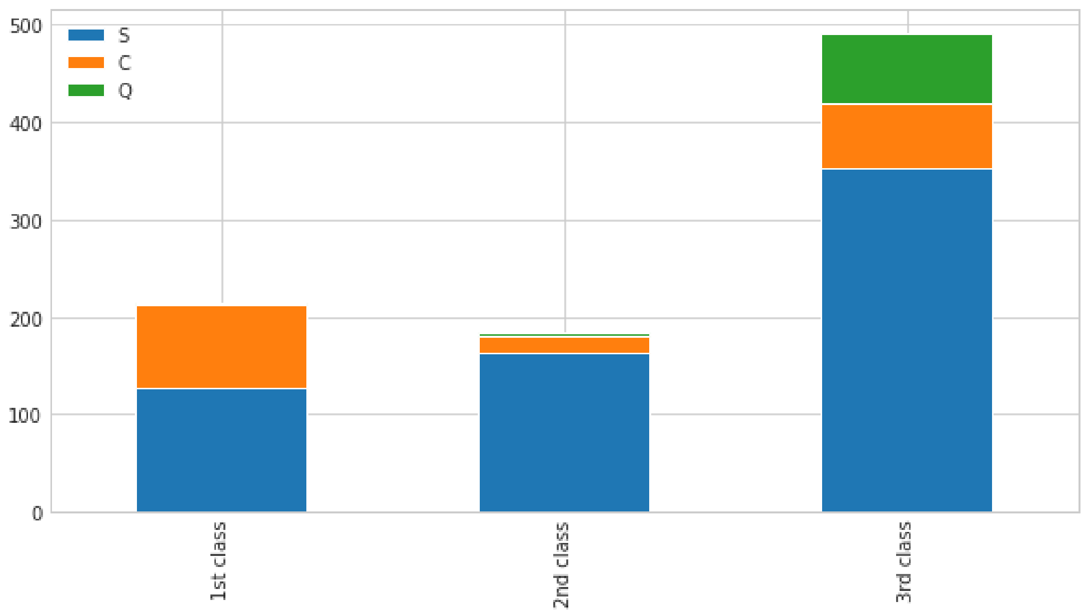
EMBARKED MAPPING

In [24]:

```
Pclass1 = train_data[train_data['Pclass']==1]['Embarked'].value_counts()
Pclass2 = train_data[train_data['Pclass']==2]['Embarked'].value_counts()
Pclass3 = train_data[train_data['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[24]:

```
&lt;AxesSubplot:&gt;
```



- **more than 50% of 1st class are from S embark**
- **more than 50% of 2nd class are from S embark**
- **more than 50% of 3rd class are from S embark**

> **So we can fill up using S and then mapping for each letter because it has only 2 missing values as mentioned beginning of analysis.**
>
> - **S : 0 (S : Southampton)**
> - **C : 1 (C : Cherbourg)**
> - **Q : 2 (Q : Queentown)**

In [25]:

```
for dataset in combine:
    dataset['Embarked'] = dataset['Embarked'].fillna('S')

for dataset in combine:
    dataset['Embarked'] = dataset['Embarked'].map( {'S': 0, 'C': 1, 'Q': 2} ).astype(int
)
```

AGE MISSING VALUES

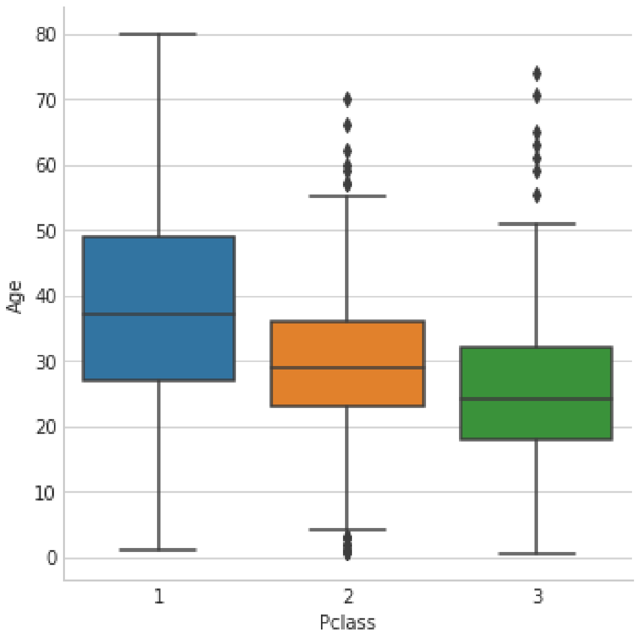
In [26]:

```
sns.factorplot(data = train_data , x = 'Pclass' , y = 'Age', kind = 'box')
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/categorical.py:3717: UserWarning: The `fac
torplot` function has been renamed to `catplot`. The original name will be removed in a f
uture release. Please update your code. Note that the default `kind` in `factorplot` (`'p
oint'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
```

Out[26]:

```
<seaborn.axisgrid.FacetGrid at 0x7fef4546bdd0>
```



> **We can fill up by their median values because there are not too much missing values**

In [27]:

```
def AgeImpute(df):
    Age = df[0]
    Pclass = df[1]

    if pd.isnull(Age):
        if Pclass == 1: return 37
        elif Pclass == 2: return 29
        else: return 24
    else:
        return Age

# Age Impute
train_data['Age'] = train_data[['Age' , 'Pclass']].apply(AgeImpute, axis = 1)
test_data['Age'] = test_data[['Age' , 'Pclass']].apply(AgeImpute, axis = 1)
```

FARE CLASS MISSING VALUE

In [28]:

```
# FARE
test_data["Fare"] = test_data["Fare"].fillna(test_data["Fare"].median())
```

> **There is just 1 missing value in test_data so median value of fare class is acceptable.**

DROP UNNECESSARY COLUMNS

In [29]:

```
features_drop = ['Cabin', 'Ticket', 'Name', 'PassengerId']
train_data.drop(features_drop, axis=1, inplace=True)
test_data.drop(features_drop, axis=1, inplace=True)
```

In [32]:

```
print("AFTER FEATURE ENGINEERING\n")
print('Train data with null values:\n', train_data.isnull().sum())
print('\nTest data with null values:\n', test_data.isnull().sum())
```

```
AFTER FEATURE ENGINEERING

Train data with null values:
 Survived       0
Pclass         0
Sex            0
Age            0
SibSp          0
Parch          0
Fare           0
Embarked       0
Title          0
family_size    0
Has_Cabin      0
dtype: int64

Test data with null values:
 Pclass         0
Sex            0
Age            0
SibSp          0
Parch          0
Fare           0
Embarked       0
Title          0
family_size    0
Has_Cabin      0
dtype: int64
```