DS 540 Python project by

Jyothsna.M

Predicting Survival rate of Titanic Ship Passengers

Importing necessary libraries for data visualization and machine learning

In [73]:

```
import pandas as pd  # Import pandas
import numpy as np   # Import numpy
import seaborn as sns # Import seaborn
import plotly.express as px  # Import plotly express for Interactive Chart
import matplotlib.pyplot as plt  # Import matplotlib
import seaborn as seabornInstance
import cufflinks as cf
cf.go_offline()
from matplotlib.animation import FuncAnimation #Import Animation Function
from sklearn.svm import SVR  # Import SVR model
from sklearn.model_selection import train_test_split # train test split
from sklearn.linear_model import LinearRegression #Import Linear Regression model
from sklearn.preprocessing import StandardScaler # Import StandardScaler
from sklearn.tree import DecisionTreeRegressor  # Import Decision Tree Regression model
from sklearn import metrics
from sklearn.model_selection import cross_val_predict  # For K-Fold Cross Validation
from sklearn.metrics import mean_squared_error  # For MSE
from math import sqrt  # For squareroot operation
from sklearn.preprocessing import PolynomialFeatures# Prediction with Polynomial
from plotly.offline import iplot, init_notebook_mode # Standard plotly imports
init_notebook_mode()
init_notebook_mode(connected=True)

%matplotlib inline
```

Explonatory data analysis

Previewing Titanic dataset

In [74]:

```
myData = pd.read_csv('C:\\Users\\15512\\OneDrive\\Desktop\\titanic.csv')# read the dataset
```

viewing first five rows of data

In [75]:

```
myData.head()# Used .head() to get the first five raws of the dataset
```

Out[75]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	0	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	1	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	0	35.0	0	0	373450	8.0500	NaN	S

Cleaning the data

In [76]:

```
myData.drop(["Name","Ticket","Cabin","Fare"], axis=1, inplace=True)
myData.head()# Removing unnecessary rows and columns
```

Out[76]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Embarked
0	1	0	3	0	22.0	1	0	S
1	2	1	1	1	38.0	1	0	C
2	3	1	3	1	26.0	0	0	S
3	4	1	1	1	35.0	1	0	S
4	5	0	3	0	35.0	0	0	S

Converting categorical column into numeric

In [77]:

```
myData['Embarked'] = myData['Embarked'].replace({"S":0,"C":1,"Q":2}) # Convert the categorical data
"Embarked" to numeric value
```

In [78]:

```
myData.head()# Previewing after converting
```

Out[78]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Embarked
0	1	0	3	0	22.0	1	0	0.0
1	2	1	1	1	38.0	1	0	1.0
2	3	1	3	1	26.0	0	0	0.0
3	4	1	1	1	35.0	1	0	0.0
4	5	0	3	0	35.0	0	0	0.0

Checking for any missing values

In [79]:

```
myData.isnull().any() #  missing values in Embarked
```

Out[79]:

```
PassengerId    False
Survived       False
Pclass         False
Sex            False
Age            False
SibSp          False
Parch          False
Embarked        True
dtype: bool
```

In [80]:

```
myData.isnull().sum() # 2 missing values
```

Out[80]:

```
PassengerId    0
Survived       0
Pclass         0
Sex            0
Age            0
SibSp          0
Parch          0
Embarked       2
dtype: int64
```

Checking for missing values,rows,columns,unique values anf feautures

In [81]:

```
print ("Rows     : " , myData.shape[0])
print ("Columns  : " , myData.shape[1])
print ("\nFeatures : \n" , myData.columns.tolist())
print ("\nMissing values :  ", myData.isnull().sum().values.sum())
print ("\nUnique values :  \n",myData.nunique())
```

```
Rows     :  891
Columns  :  8

Features :
 ['PassengerId', 'Survived', 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Embarked']

Missing values :   2

Unique values :
 PassengerId    891
Survived         2
Pclass           3
Sex              2
Age             89
SibSp            7
Parch            7
Embarked         3
dtype: int64
```

In [82]:

```
myData.dtypes # Identification of data types
```

Out[82]:

```
PassengerId      int64
Survived         int64
Pclass           int64
Sex              int64
Age            float64
SibSp            int64
Parch            int64
Embarked       float64
dtype: object
```

In [83]:

```
myData.shape #Used .shape to find the size of the dataset
```

Out[83]:

```
(891, 8)
```

In [84]:

```python
myData.describe() # Description of variables
```

Out[84]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Embarked
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000	889.000000
mean	446.000000	0.383838	2.308642	0.352413	29.699293	0.523008	0.381594	0.362205
std	257.353842	0.486592	0.836071	0.477990	13.002015	1.102743	0.806057	0.636157
min	1.000000	0.000000	1.000000	0.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	0.000000	22.000000	0.000000	0.000000	0.000000
50%	446.000000	0.000000	3.000000	0.000000	29.700000	0.000000	0.000000	0.000000
75%	668.500000	1.000000	3.000000	1.000000	35.000000	1.000000	0.000000	1.000000
max	891.000000	1.000000	3.000000	1.000000	80.000000	8.000000	6.000000	2.000000

Previewing info of variables like int,float and object

In [85]:

```python
myData.info() # get info
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 8 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
Embarked       889 non-null float64
dtypes: float64(2), int64(6)
memory usage: 55.8 KB
```

Data visualization

Find out total no of male/female passengers

In [86]:

```python
myData.Sex.value_counts() # Sex Value Counts
```

Out[86]:

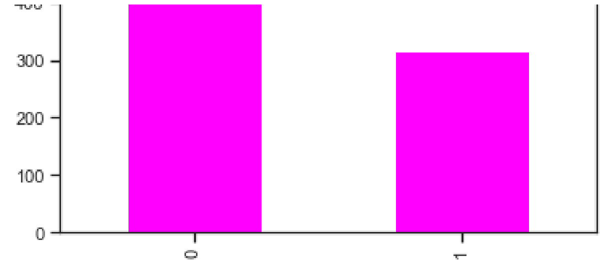
```
0    577
1    314
Name: Sex, dtype: int64
```

In [87]:

```python
myData.Sex.value_counts().plot(kind='bar',color='magenta') # Bar Plot of sex value
```

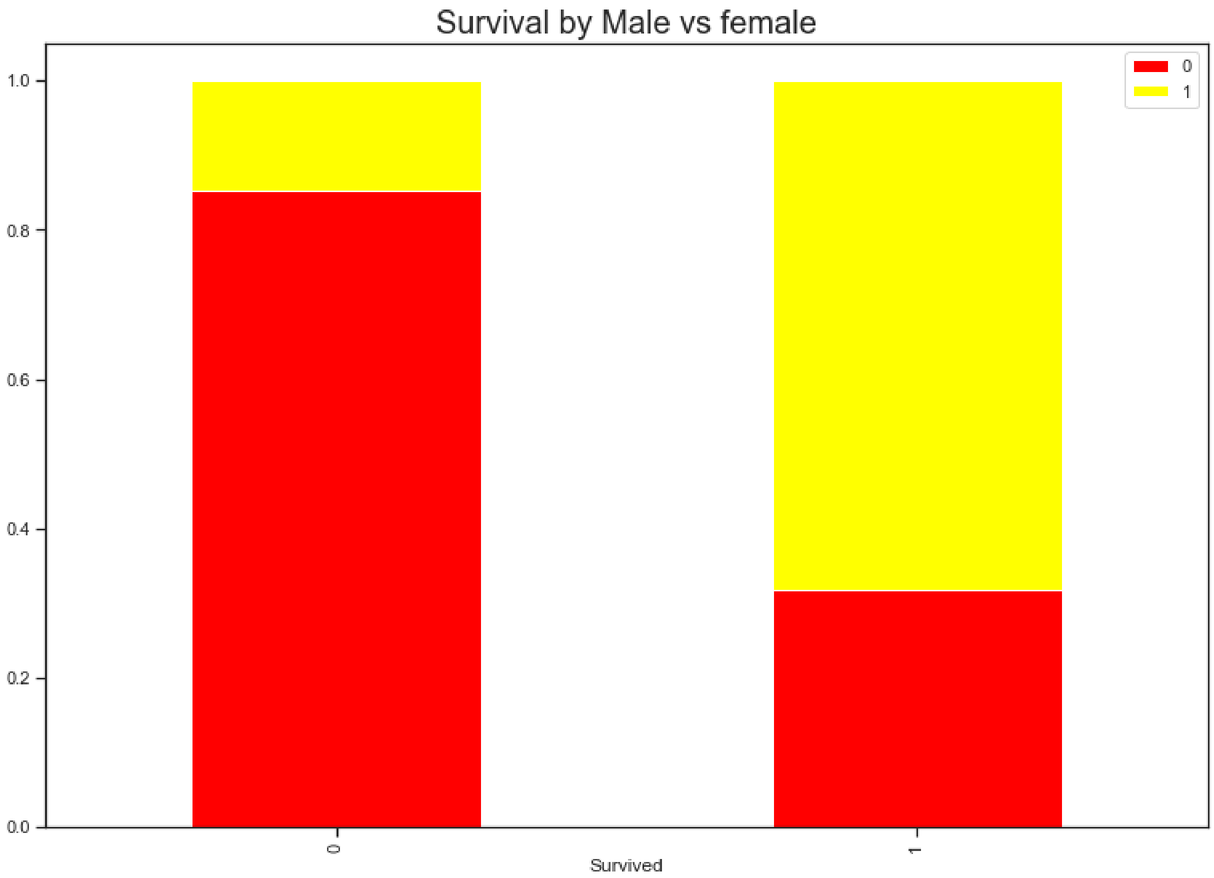
Out[87]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x239083e3b38>
```



In [88]:

```
# Stacked bar plot of Survival ratio of males and females
plt.rcParams['figure.figsize'] = (13, 9)
Y = pd.crosstab(myData['Survived'], myData['Sex'])
Y.div(Y.sum(1).astype(float), axis = 0).plot(kind = 'bar', stacked = True,color=['red','yellow'])
plt.title('Survival by Male vs female', fontweight = 30, fontsize = 20)
plt.legend(loc="upper right")
plt.show()
```



- The first fig above shows that males are more (577) than females (314) travelling in the ship
- The sec fig above shows that females(1) survival ratio is high than males (0).Yellow color is survived and red is represented by not survived.

Find total number of passengers in each passenger class

In [93]:

```
myData.Pclass.value_counts()# Pclas value counts
```

Out[93]:

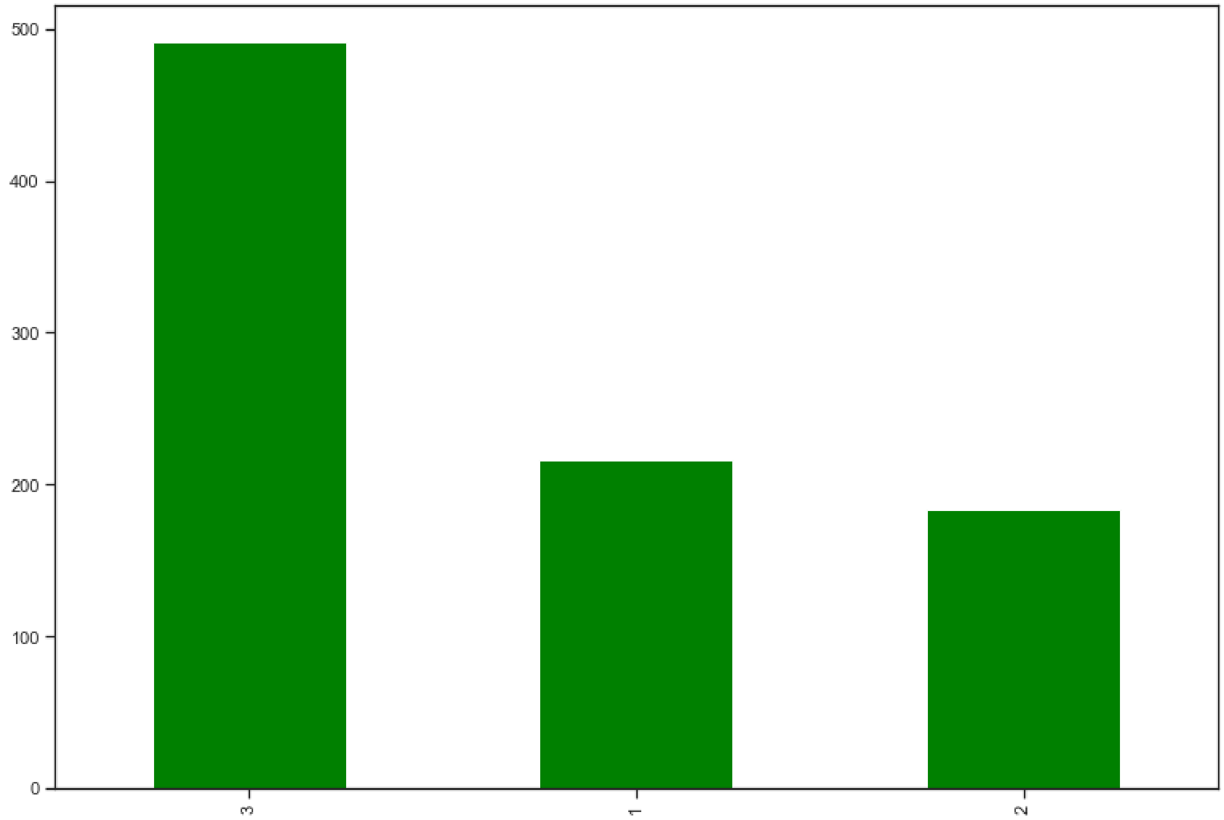
```
3    491
1    216
2    184
Name: Pclass, dtype: int64
```

In [94]:

```
myData.Pclass.value_counts().plot(kind='bar',color='green')# Bar Plot of Pclass count
```

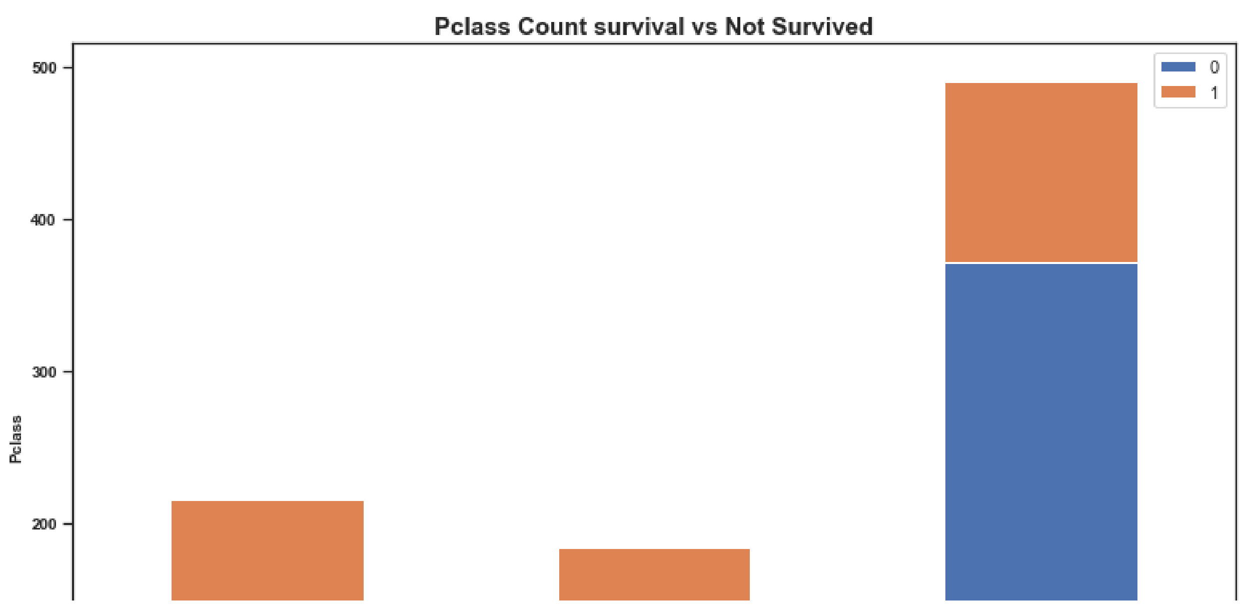
Out[94]:

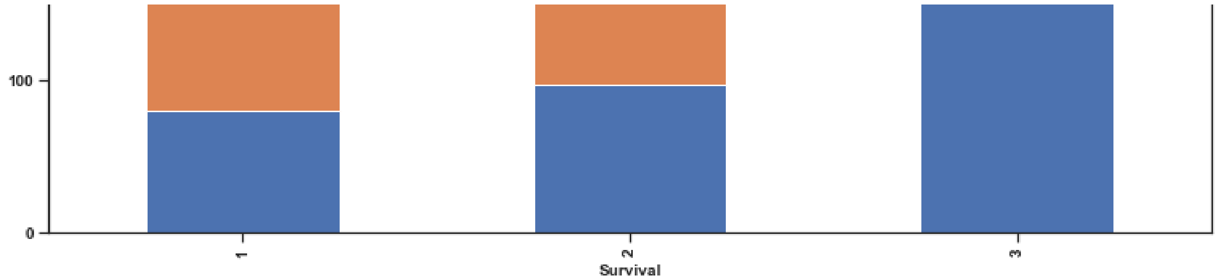
<matplotlib.axes._subplots.AxesSubplot at 0x239084e6780>



In [95]:

```
# Stacked Bar plot of Pclass Survived vs not survived
loc_plt=pd.crosstab(myData['Pclass'],myData['Survived'])
loc_plt.plot(kind='bar',stacked=True);
plt.title('Pclass Count survival vs Not Survived',fontsize=15,fontweight='bold')
plt.ylabel('Pclass',fontsize=10,fontweight='bold')
plt.xlabel('Survival',fontsize=10,fontweight='bold')
plt.xticks(fontsize=10,fontweight='bold')
plt.yticks(fontsize=10,fontweight='bold');
plt.legend();
```





- The first fig above shows that more number pf passengers (491) are travelling in 3rd class,216 are travelling in 1st calss and least number(184) passengers are travelling in 2nd class
- The second fig above shows that survival rate is high in Pclass 1 and survival rate is almost equal in Pclass2 and Pclass1 Orange color is survived and blue color represented by not survived

Find out total number of survived/not survived passengers

In [96]:

```
myData.Survived.value_counts() # Survived Value count
```

Out[96]:

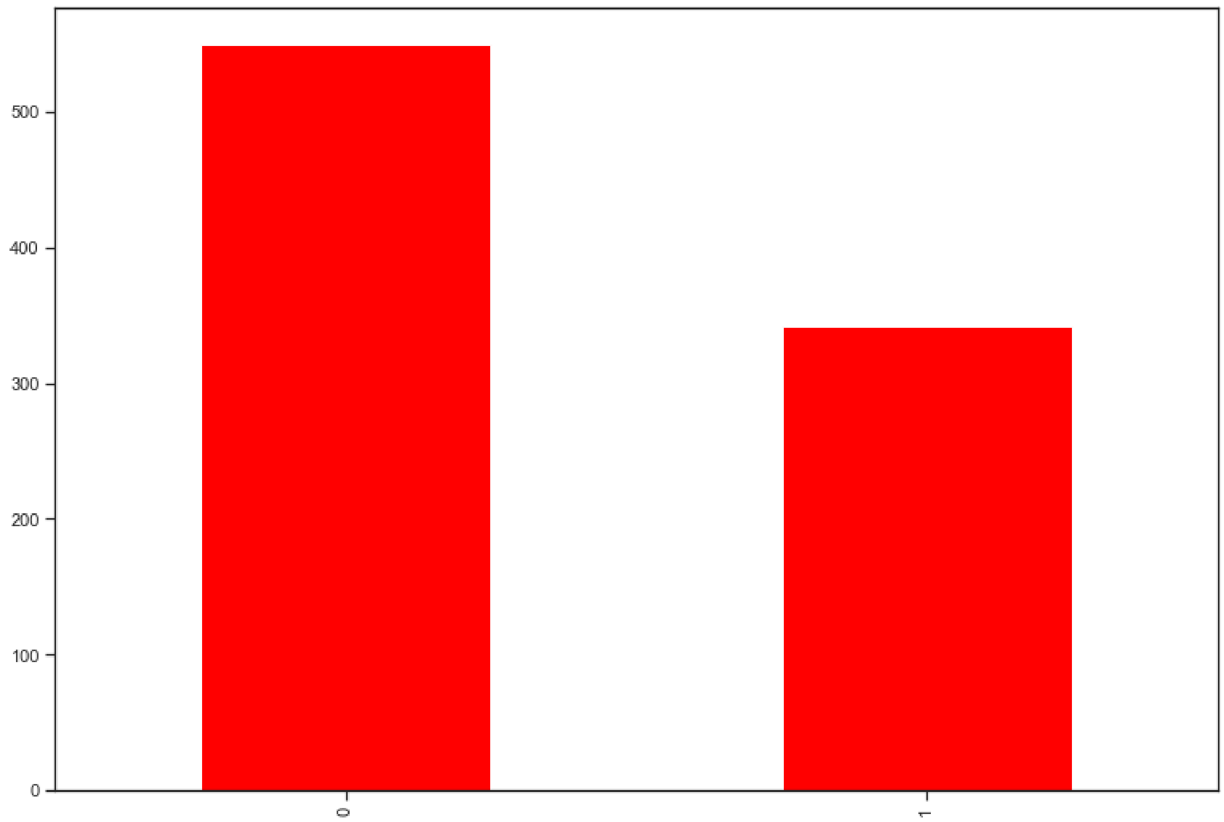
```
0    549
1    342
Name: Survived, dtype: int64
```

In [97]:

```
myData.Survived.value_counts().plot(kind='bar',color='red')# Bar plot of Survived count
```

Out[97]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x23908a8c198>
```



- The first fig above shows that survived ratio is less (342) when compared to not survived (549)

Find total number of passengers by point of Embarkation

In [98]:

```
myData.Embarked.value_counts()# Embarked value count
```

Out[98]:

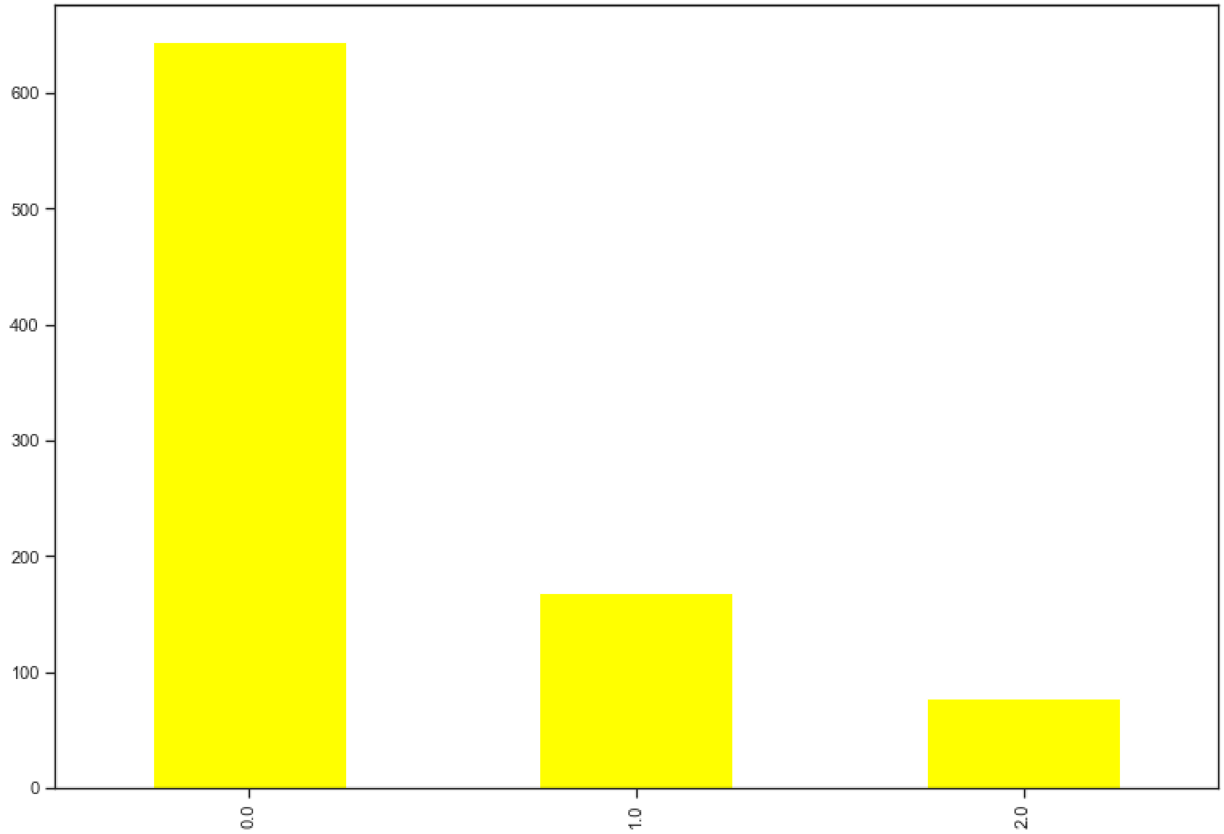
```
0.0    644
1.0    168
2.0     77
Name: Embarked, dtype: int64
```

In [99]:

```
myData.Embarked.value_counts().plot(kind='bar',color='yellow')# Bar plot of embarked value count
```

Out[99]:

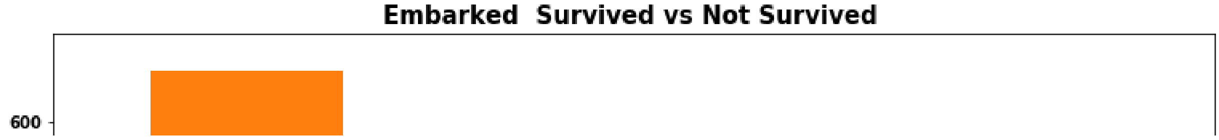
```
<matplotlib.axes._subplots.AxesSubplot at 0x23908563d68>
```

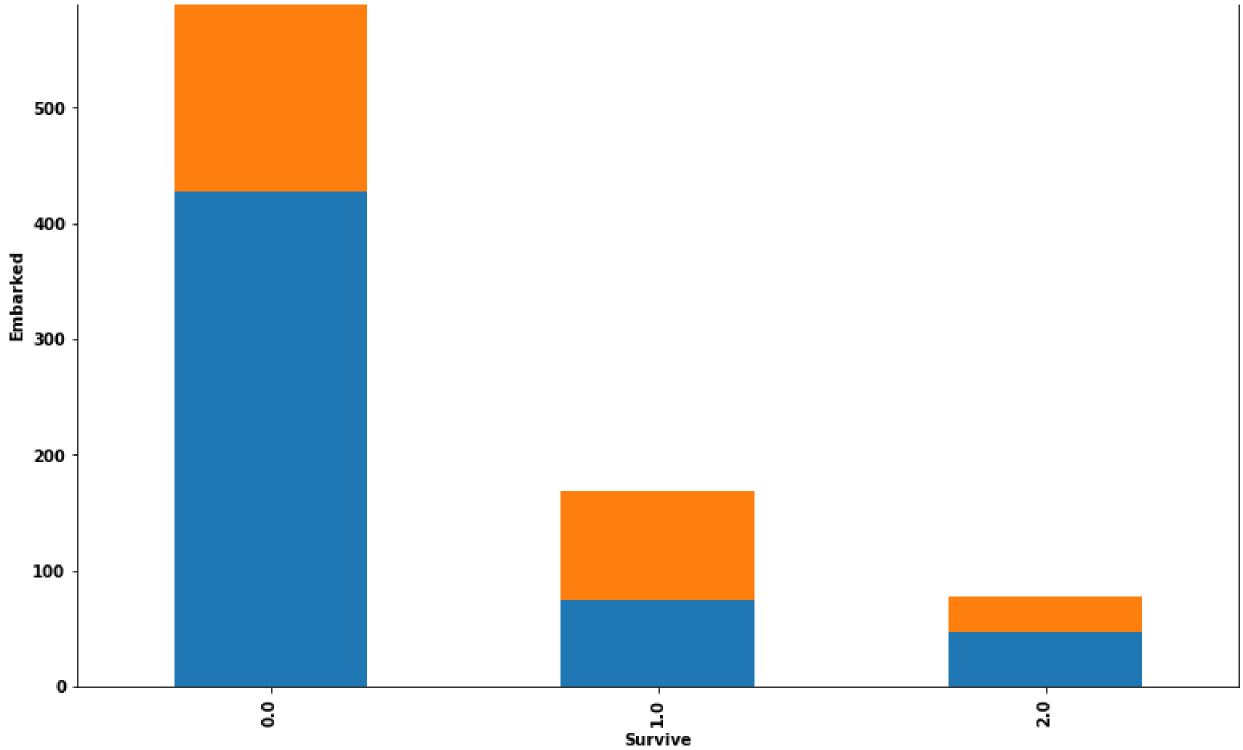


In [23]:

```
# Stacked bar plot of Embarked survived vs not survived
loc_plt=pd.crosstab(myData['Embarked'],myData['Survived'])
loc_plt.plot(kind='bar',stacked=True);
plt.title('Embarked  Survived vs Not Survived ',fontsize=15,fontweight='bold')
plt.ylabel('Embarked',fontsize=10,fontweight='bold')
plt.xlabel('Survive',fontsize=10,fontweight='bold')
plt.xticks(fontsize=10,fontweight='bold')
plt.yticks(fontsize=10,fontweight='bold');
plt.legend().remove();
```

Embarked Survived vs Not Survived





- The first fig above shows that More number of passengers are from southhampston(644),the count of passengers from cherbourg are 168 and least number of passengers are from Queenstown(77)
- The sec fig above shows that survived ratio is more in passengers from southhampston,next is cherbourg and least is queenstown.

Find out total number of passengers of different age groups (0-30,31-60,60-80 and greater than 80 >80)

In [100]:

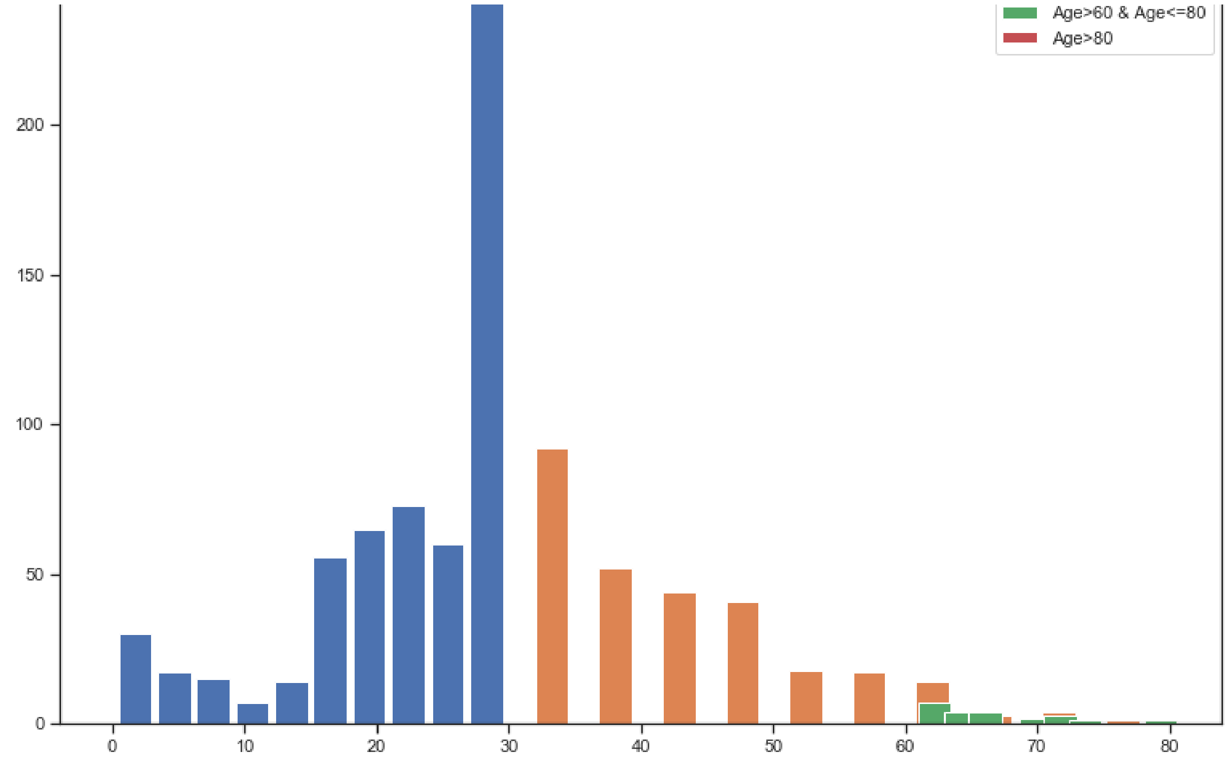
```
# Count of Passengers different age group
Ag1=myData.Age[myData.Age<=30]
Ag2=myData.Age[myData.Age>31]
Ag3=myData.Age[(myData.Age>60) &(myData.Age<=80)]
Ag4=myData.Age[myData.Age>80]
print(Ag1.size)
print(Ag2.size)
print(Ag3.size)
print(Ag4.size)
```

```
586
286
22
0
```

Histogram of Number of passengers in each age group

In [101]:

```
# Histogram of different age group
plt.hist(Ag1,width=2.5,label='Age<=30')
plt.hist(Ag2,width=2.5,label='Age>31')
plt.hist(Ag3,width=2.5,label='Age>60 & Age<=80')
plt.hist(Ag4,width=2.5,label='Age>80')
plt.legend()
plt.show()
```

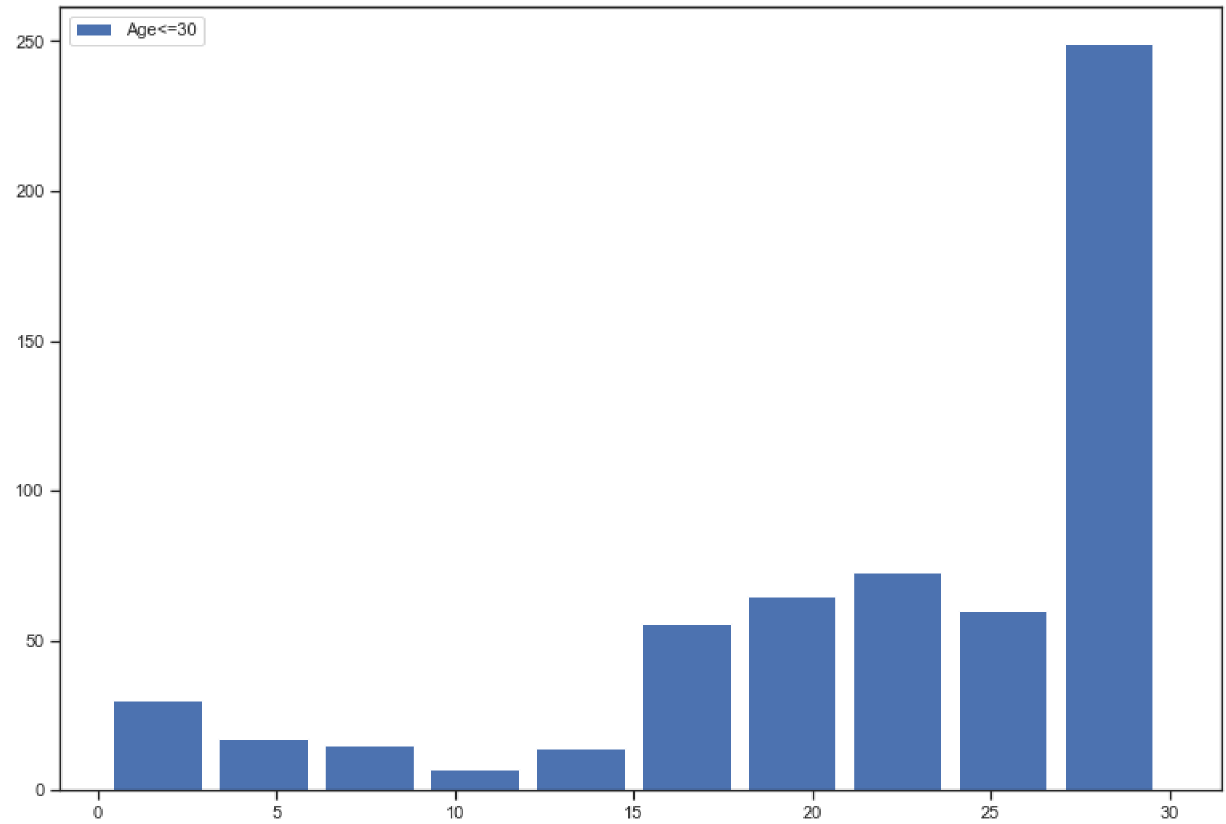


In [102]:

```
# Histogram of Ag1
plt.hist(Ag1,width=2.5,label='Age<=30')
plt.legend()
```

Out[102]:

```
<matplotlib.legend.Legend at 0x23908483780>
```



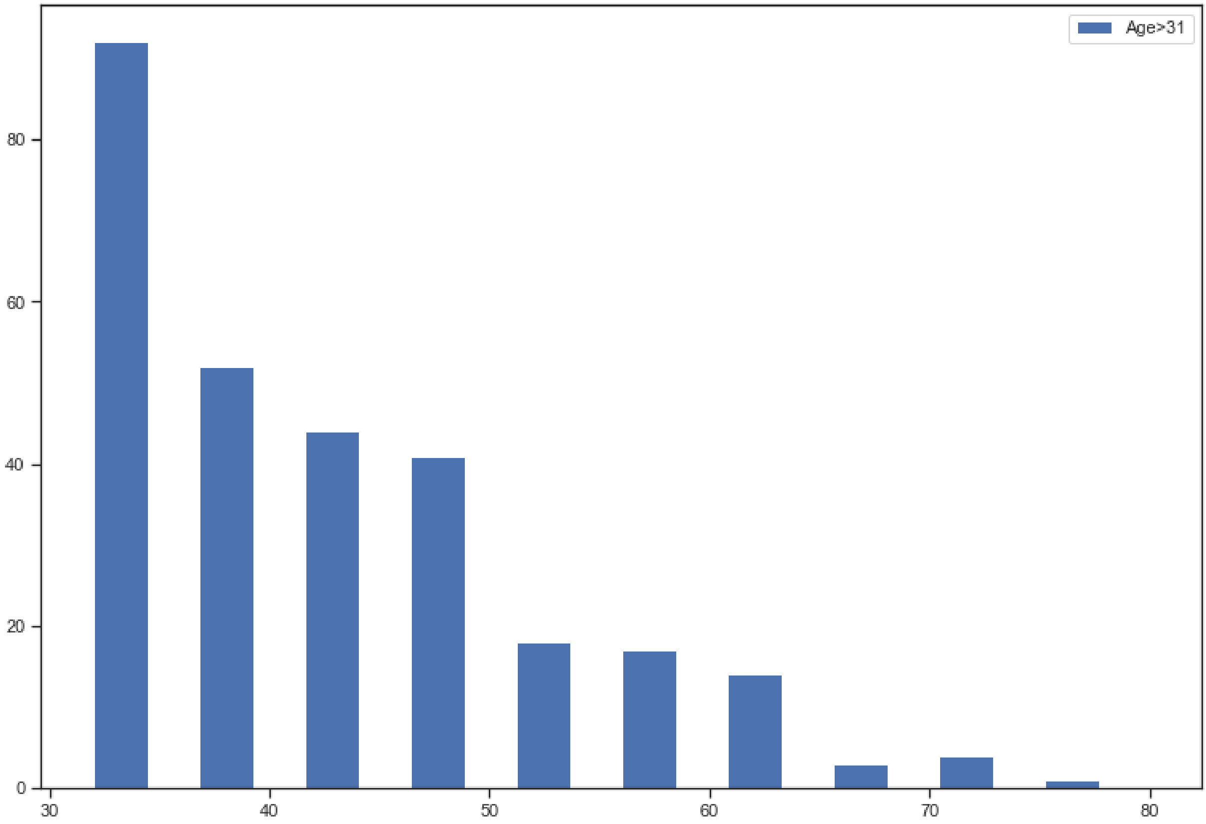
In [103]:

```
# Histogram of Ag2
plt.hist(Ag2,width=2.5,label='Age>31')
```

```
plt.hist(Ag2,width=2.5,label='Age>31')
plt.legend()
```

Out[103]:

```
<matplotlib.legend.Legend at 0x23907639ba8>
```

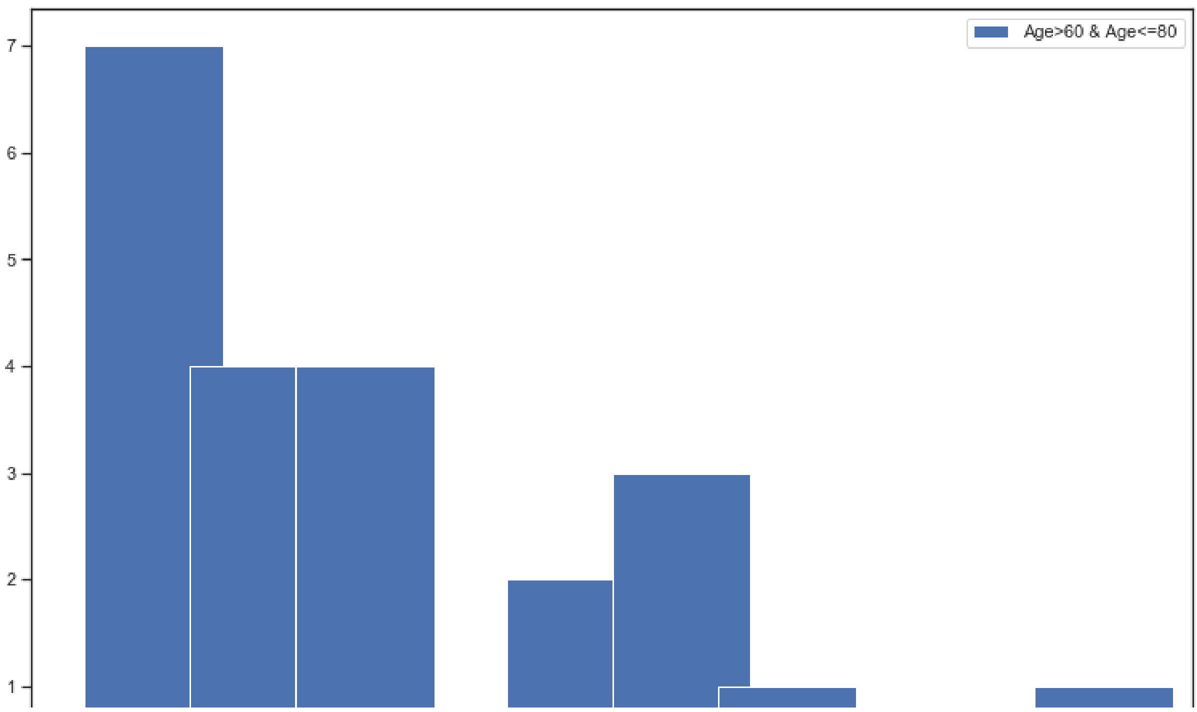


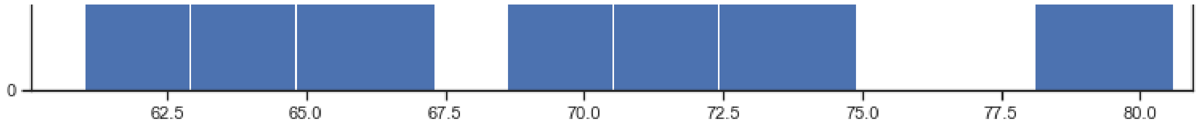
In [104]:

```
# Histogram of Ag3
plt.hist(Ag3,width=2.5,label='Age>60 & Age<=80')
plt.legend()
```

Out[104]:

```
<matplotlib.legend.Legend at 0x23905ad7fd0>
```





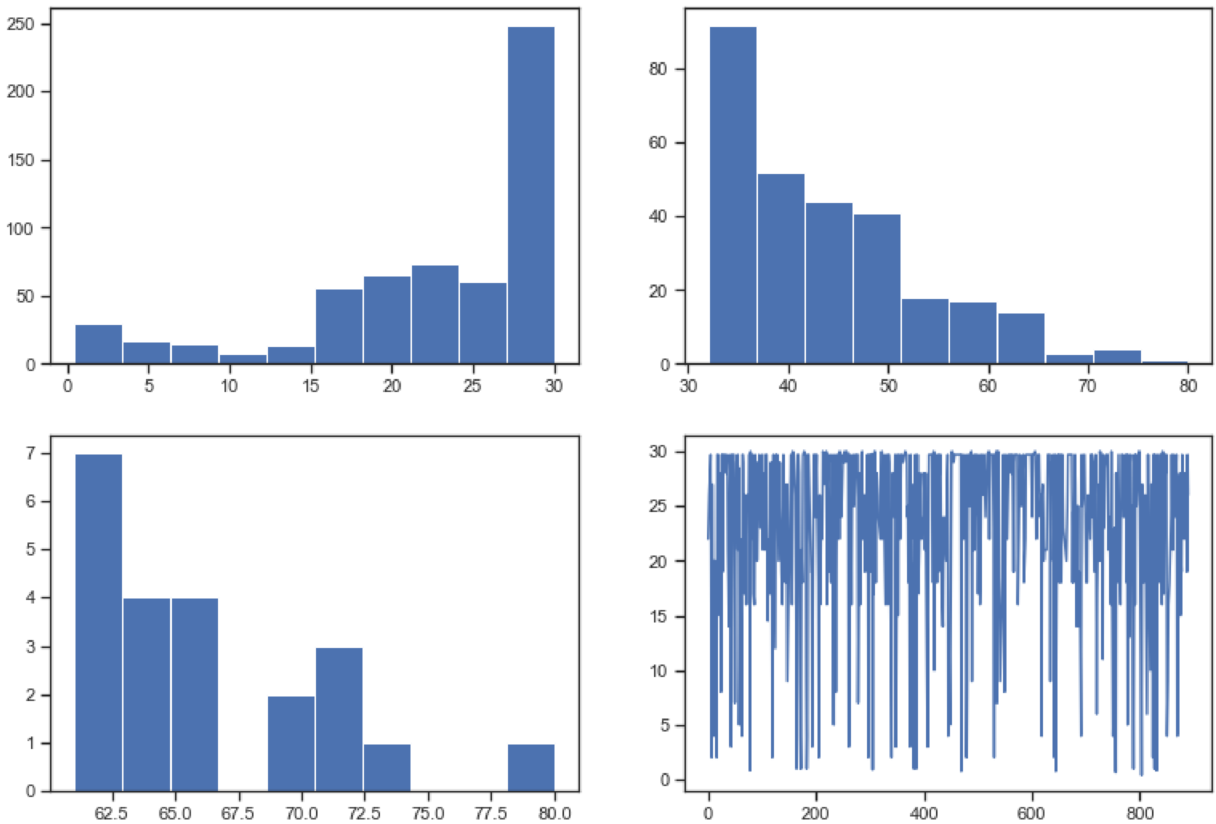
- The above figs shows that more number of passengers are in the age group 25-30 and least number of passenger in the age gropup from 70 and above
- Infants below 1 yr are the youngest passengers and passengers above 70 are the oldest passengers travelling in the ship

In [105]:

```
# Sub Plots of Ag1,Ag2,Ag3
plt.subplot(2,2,1)
plt.hist(Ag1)
plt.subplot(2,2,2)
plt.hist(Ag2)
plt.subplot(2,2,3)
plt.hist(Ag3)
plt.subplot(2,2,4)
plt.plot(Ag1)
plt.show
```

Out[105]:

```
<function matplotlib.pyplot.show(*args, **kw)>
```

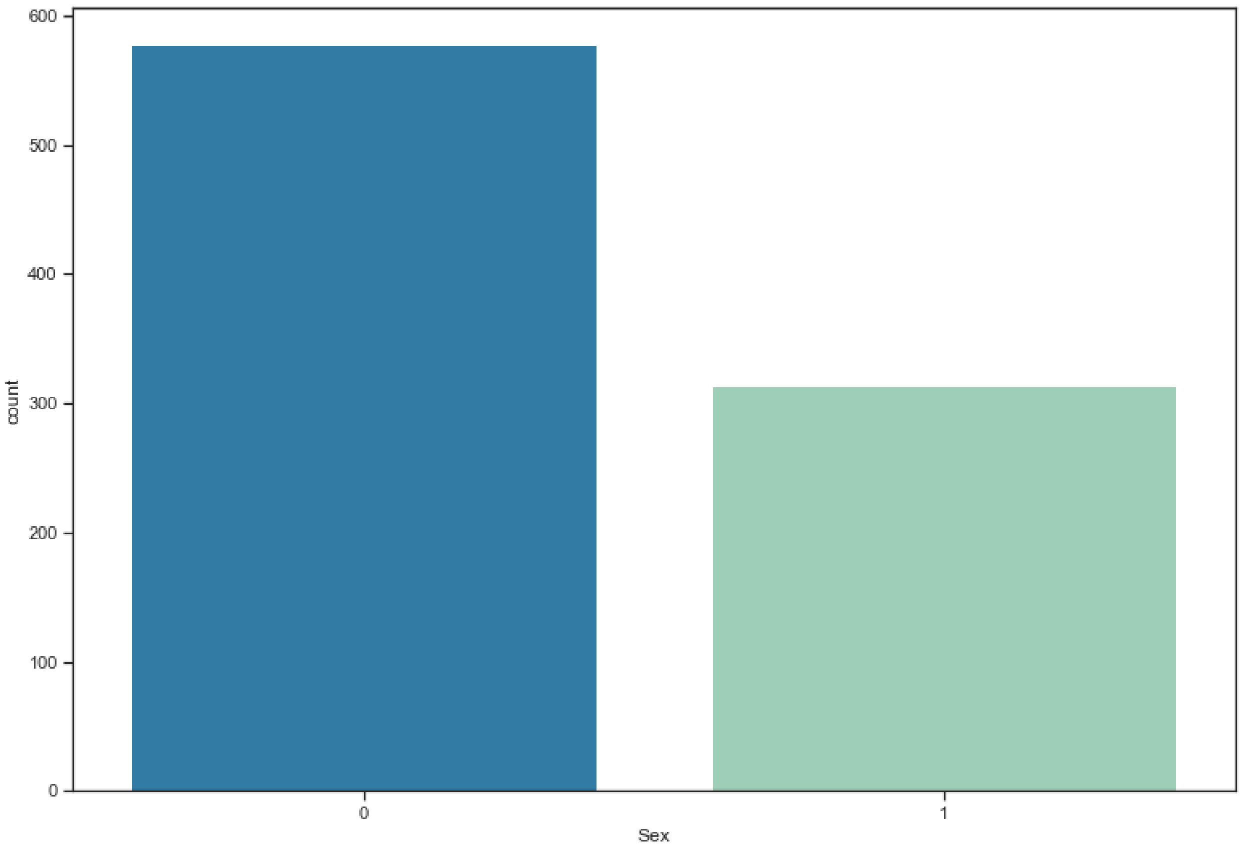


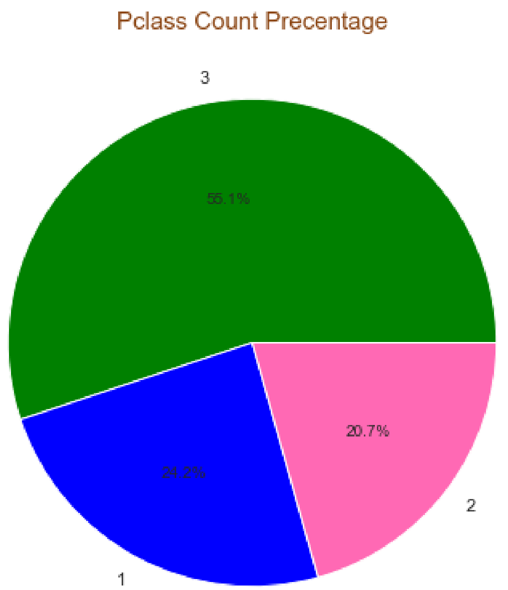
Histogram,Pie Chart and swarm plot

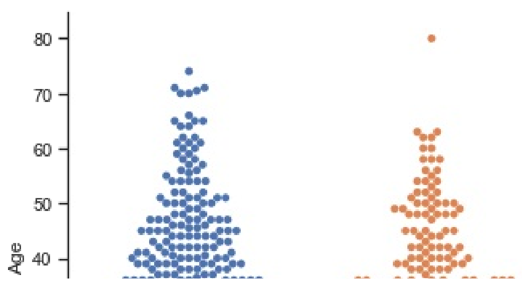
In [106]:

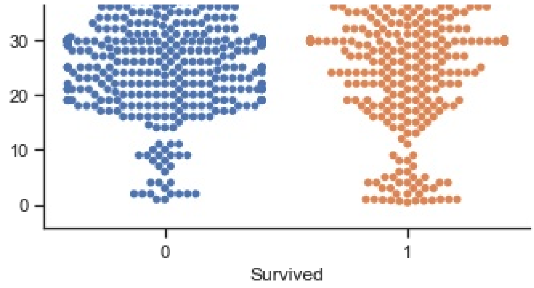
```
# Simple Histogram
sns.countplot(x="Sex", data=myData ,palette = 'YlGnBu_r'); # Sex Count
plt.show() # plot the histogram

# Pie Chart
labels=myData.Pclass.value_counts().index # Pclass count percentage
colors=["green","blue","hotpink","yellow","navy","#9b59b6"] # color of pie chart
sizes=myData.Pclass.value_counts().values
plt.figure(figsize=(7,7)) #plot the figure
```

```
plt.figure(figsize=(7,7)) #plot the figure
plt.pie(sizes,labels=labels,colors=colors,autopct="%1.1f%%")
plt.title("Pclass Count Precentage",color="saddlebrown",fontsize=15) #title of pie chart

# Swarm Plot
# analyzing the Survived by Age
sns.catplot(x="Survived", y="Age", kind="swarm", data=myData);
```









- The first fig above shows that males are more than females
- The sec pie chart shows that 55% are travelling in 3rd class,24% are travelling in 1st class and 20% are travelling in 2nd class
- The third swarm plot above shows that in both male and females survived passengers are more in the age group between 20-30

Interactive Histogram

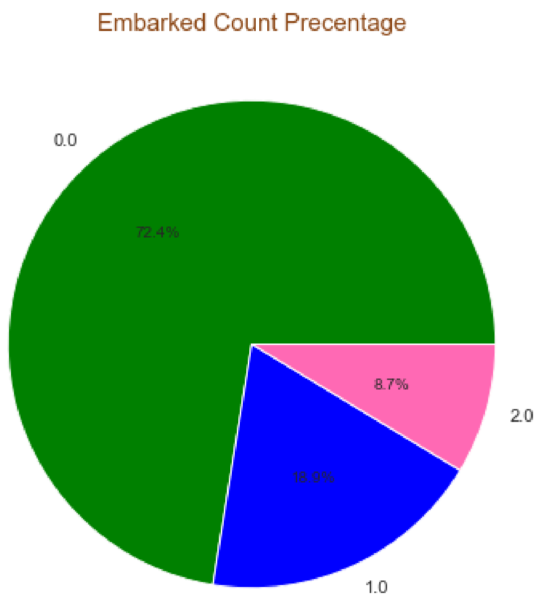
Pie chart and bar chart,countplot

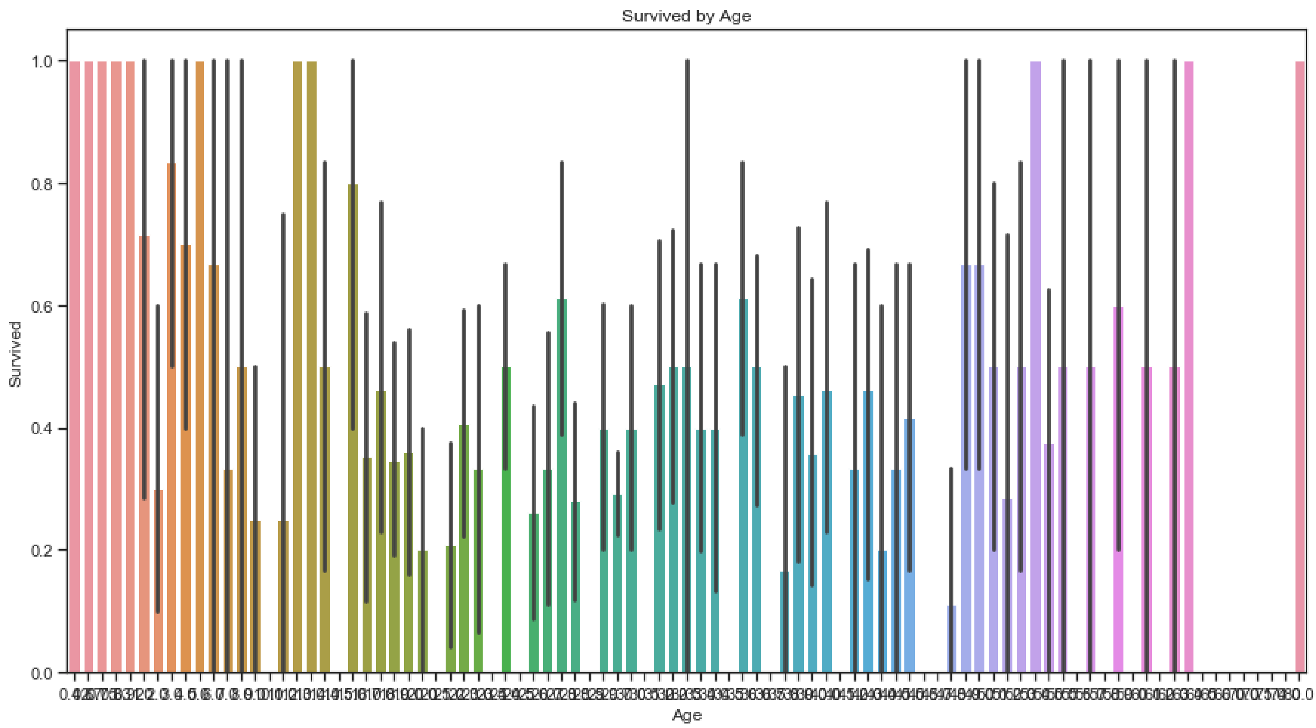
In [107]:

```
# simple interactive Histogram
fig = px.histogram(myData, x="Age") # Age count of different age groups
fig.show() # plot the interactive histogram for Age

# Pie Chart
labels=myData.Embarked.value_counts().index # Compare the Embarked value counts
colors=["green","blue","hotpink","yellow","navy","#9b59b6"] # color of pie chart
sizes=myData.Embarked.value_counts().values
plt.figure(figsize=(7,7)) #plot the figure
plt.pie(sizes,labels=labels,colors=colors,autopct="%1.1f%%")
plt.title("Embarked Count Precentage",color="saddlebrown",fontsize=15) #title of pie chart

plt.figure(figsize=(15,8)) #figure size
ax = sns.barplot(x='Age', y='Survived', data=myData)# simple barplot Survived by Age
ax.set_title('Survived by Age') #title for barplot
```

Out[107]:

```
Text(0.5, 1.0, 'Survived by Age')
```





- The first fig above shows that count of passengers between 28-29 is 224,1 passenger is travelling in age of 80,5 passengers travelling in the age of 70-71 14 are travelling in age group of 0-2 yrs old.
- The sec pie chart shows that 72% passengers point of embarkation is southhampston,cherbourg is 18% and 8.7% are from queenstown
- The third fig above shows survival rate in each age group

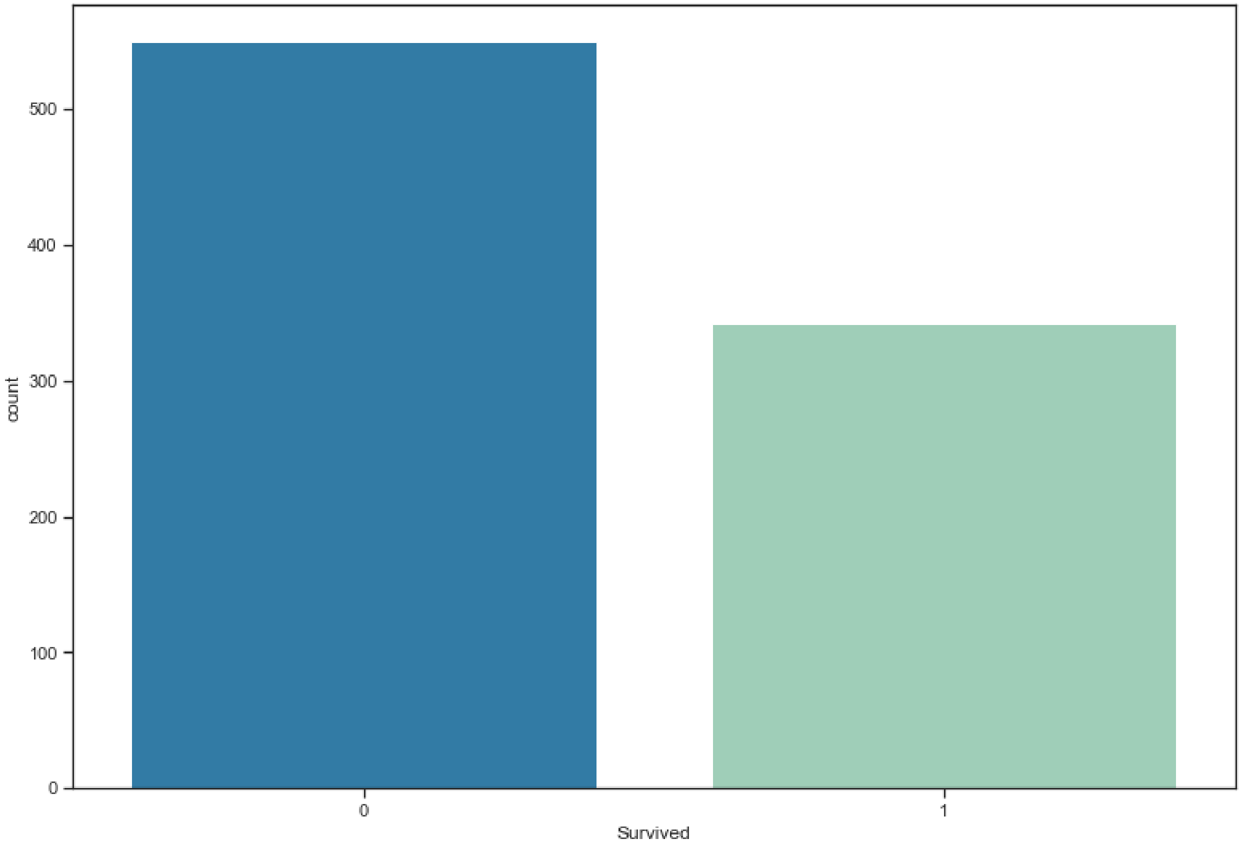
Histogram & Violin plot

In [108]:

```
vc = myData["Survived"].value_counts() # count Survived
print(vc)

# Simple Histogram for Survived
sns.countplot(x="Survived", data=myData ,palette = 'YlGnBu_r'); # smoker vs non smoker
plt.show()
```

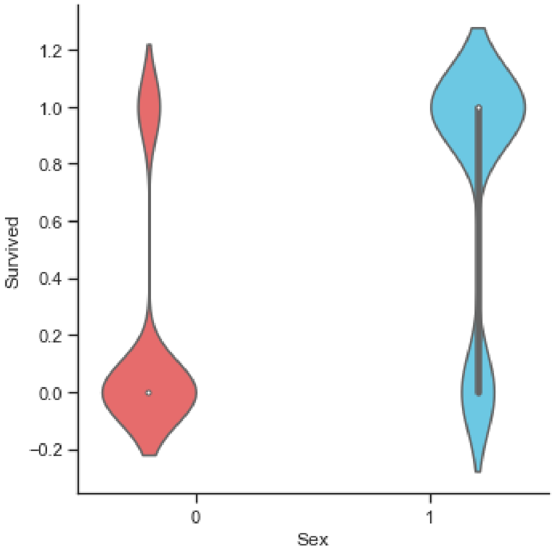
```
#Catplot combined people who survived male vs female
pal = ["#FA5858", "#58D3F7"] # palette for catplot
sns.catplot(x="Sex", y="Survived", hue="Sex",kind="violin", data=myData, palette = pal)
plt.figure(figsize=(15,8)) #figure size
```

```
0    549
1    342
Name: Survived, dtype: int64
```



Out[108]:

```
<Figure size 1080x576 with 0 Axes>
```




```
<Figure size 1080x576 with 0 Axes>
```

- The first fig above shows that not survived (549) are higher than the survived (342)
- The sec violin plot above shows that survided ratio is high in females than males

- The sec violin plot above shows that survided ratio is high in females than males

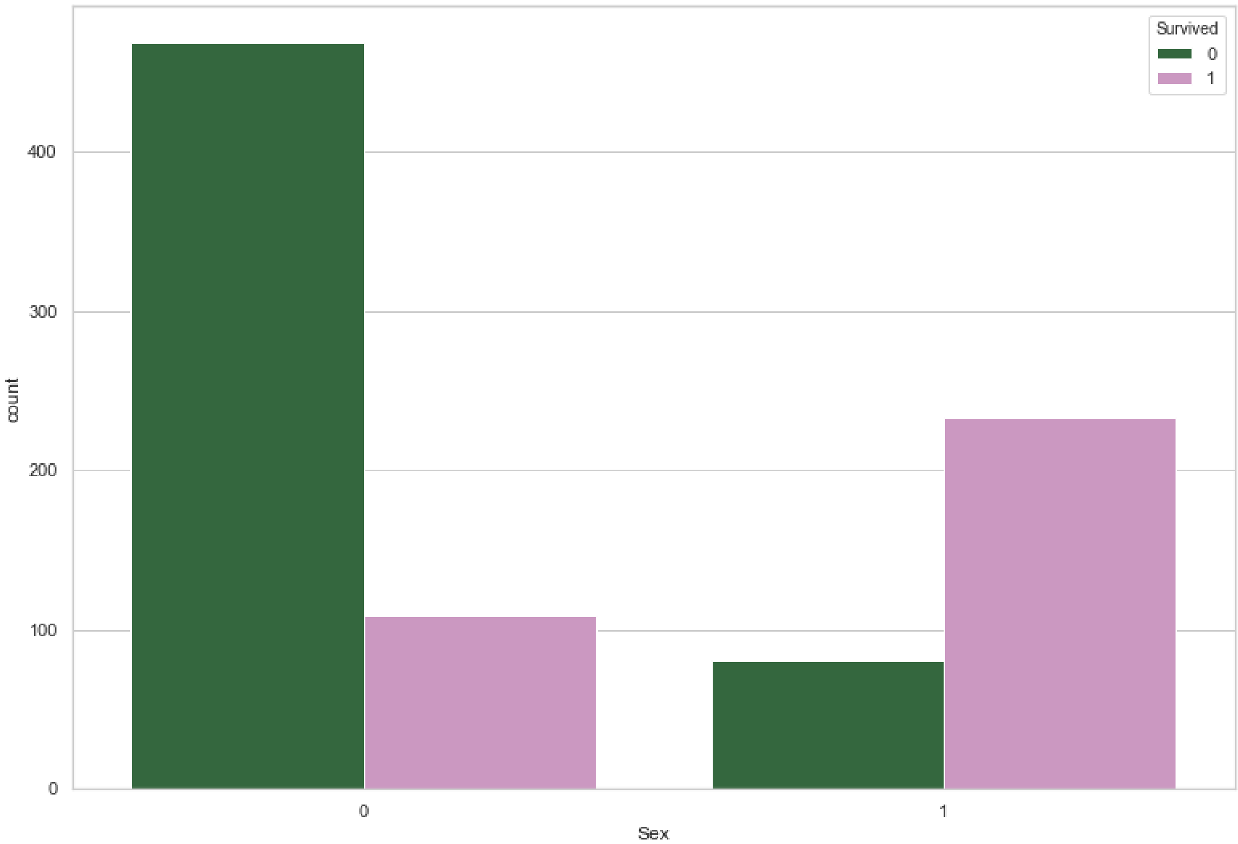
Count plot & Pie chart

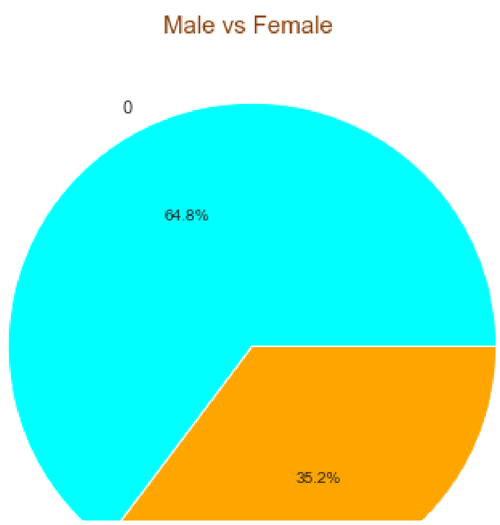
In [109]:

```
sns.set_style('whitegrid')  # using sns.countplot between Survived vs male and female
# male survivors is less than female survivors
sns.countplot(x='Sex', hue='Survived', data=myData, palette='cubehelix')


labels=myData.Sex.value_counts().index  # labels for the pie chart
sizes=myData.Sex.value_counts().values  # size for the pie chart
colors=["cyan","orange","hotpink","green","navy","#9b59b6"]# colors for pie chart
plt.figure(figsize=(7,7))
plt.pie(sizes,labels=labels,colors=colors,autopct="%1.1f%%") # plot the pie chart
plt.title("Male vs Female ",color="saddlebrown",fontsize=15)# plot the title
```

Out[109]:

```
Text(0.5, 1.0, 'Male vs Female ')
```



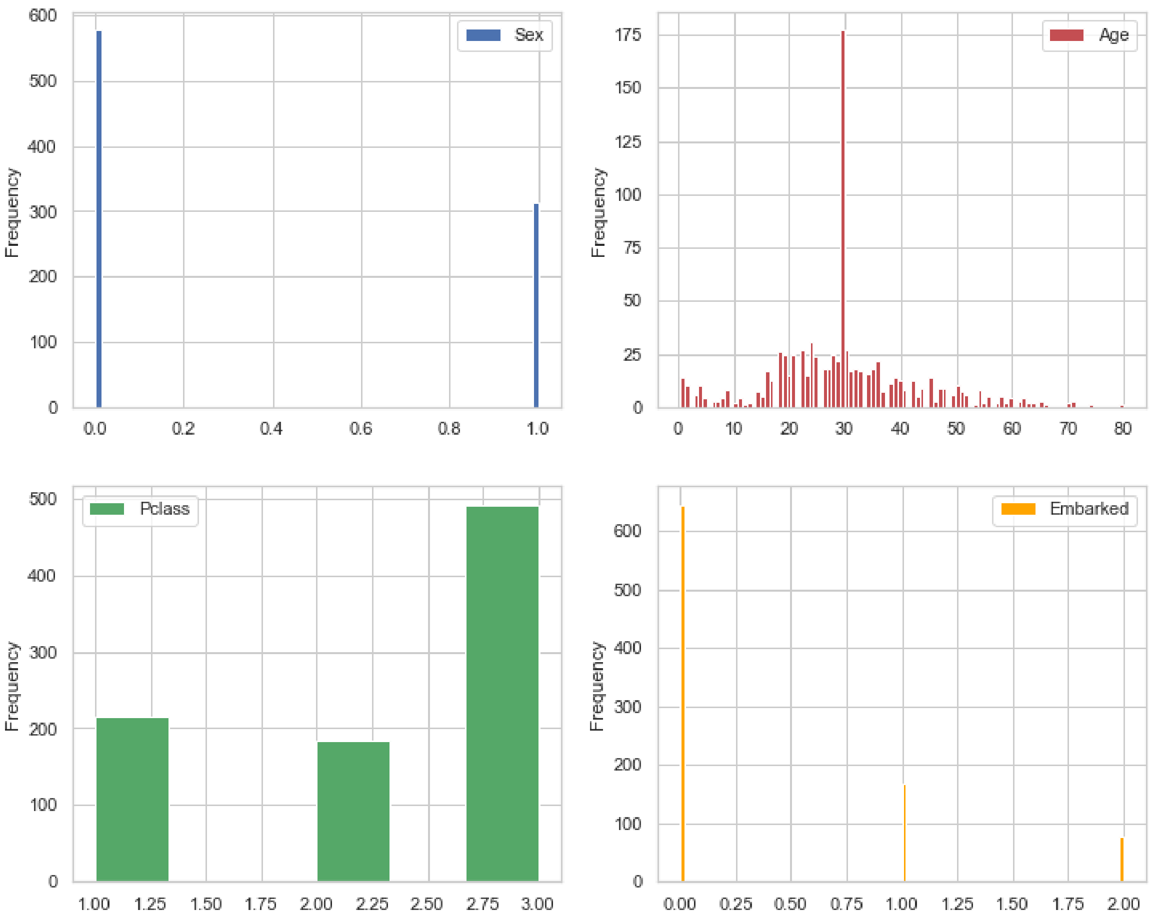


- The first fig shows that survived ratio is high in females and not survived ratio is high in males
- The sec fig shows that 64.8% are males and 36.2% are females travelling in the ship

Histogram of count of passengers in each variable

In [110]:

```
# Histogram of different variables
fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(12, 10))
myData.plot(kind="hist", y="Sex", bins=70, color="b", ax=axes[0][0]) # plot the frequency of Sex
myData.plot(kind="hist", y="Age", bins=100, color="r", ax=axes[0][1]) # plot the frequency of Age
myData.plot(kind="hist", y="Pclass", bins=6, color="g", ax=axes[1][0])# plot the frequency of Pclas
s
myData.plot(kind="hist", y="Embarked", bins=100, color="orange", ax=axes[1][1])# plot the frequency
of Embarked
plt.show()# plot the graphs
```



- The above histograms shows the count in each category.
- Males are more than females
- Passengers are more in between age group 25-25
- Passengers count is more in Pclass 3
- Passengers count is more from Southhampston

Heat Map

In [111]:

```
#Get Correlation between different variables
corr =myData.corr(method='kendall')
plt.figure(figsize=(15,8))
sns.heatmap(corr, annot=True)
myData.columns
```

Out[111]:

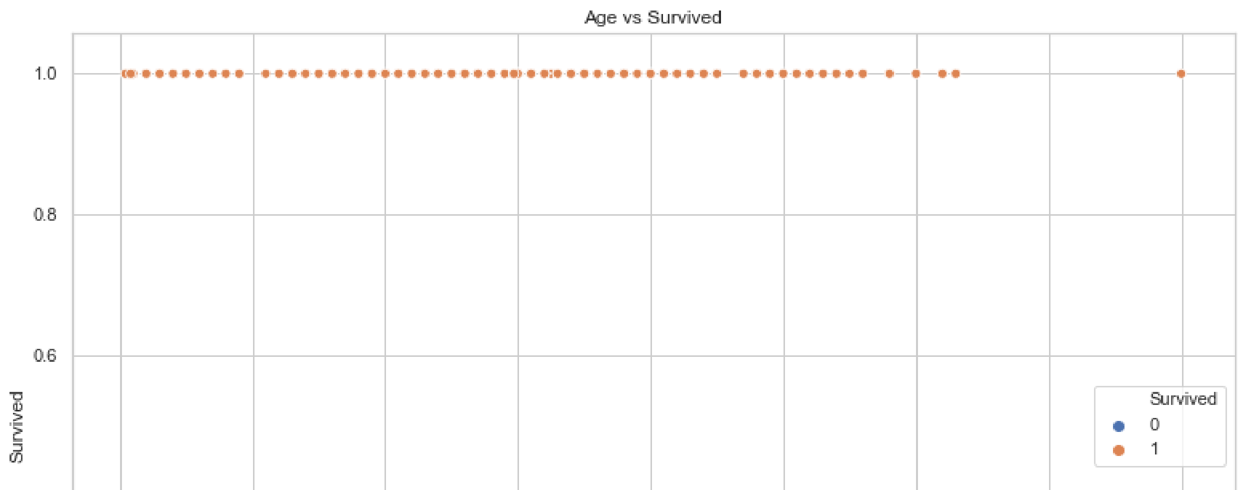
```
Index(['PassengerId', 'Survived', 'Pclass', 'Sex', 'Age', 'SibSp', 'Parch',
       'Embarked'],
      dtype='object')
```

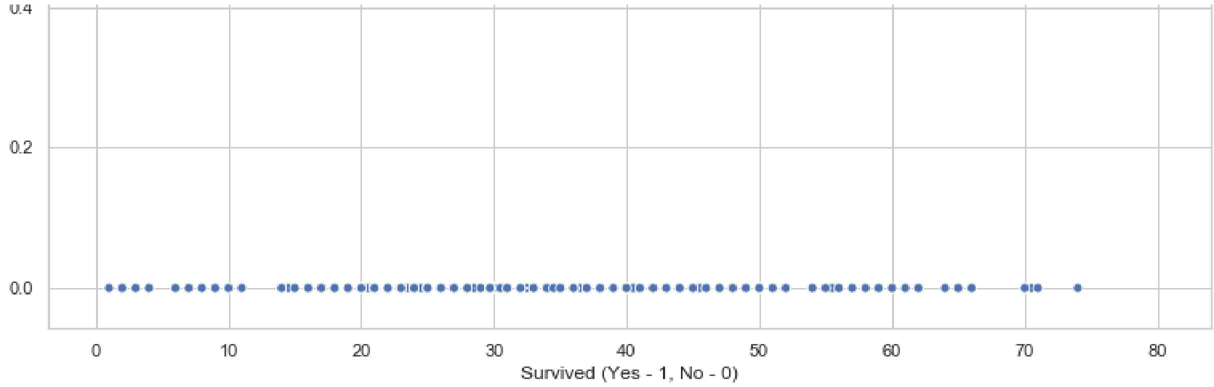
- The above heatmap shows correlation beween different variables
- There is a high correlation between Sex and Survived and Pclass and Survived,Age and Pclass.

Scatterplot

In [112]:

```
ax = sns.scatterplot(data=myData,x='Age',y='Survived',hue='Survived')# scatter plot age vs Survived
ax.set_title("Age vs Survived")# plot the title
plt.xlabel("Survived (Yes - 1, No - 0)")# plot x label
plt.ylabel("Survived")# plot y label
plt.show(ax)# plot the graph
```





- The above scatter plot shows Survived and not survived in different age groups

Interactive barplot

In [113]:

```
# Interactive barplot of Parch,SibSp vs Survived
myData.iplot(kind='bar', x=['SibSp'],y='Survived') # Interactive Bar Plot
myData.iplot(kind='bar', x=['Parch'],y='Survived') # Interactive Bar Plot
```

- The fist fig shows that passengers travellling alone survival ratio is high compared to passengers travelling with spouse and siblings
- The sec fig shows that passengers travellling alone survival ratio is high compared to travelling with parents and children

Interactive 3D plot

In [114]:

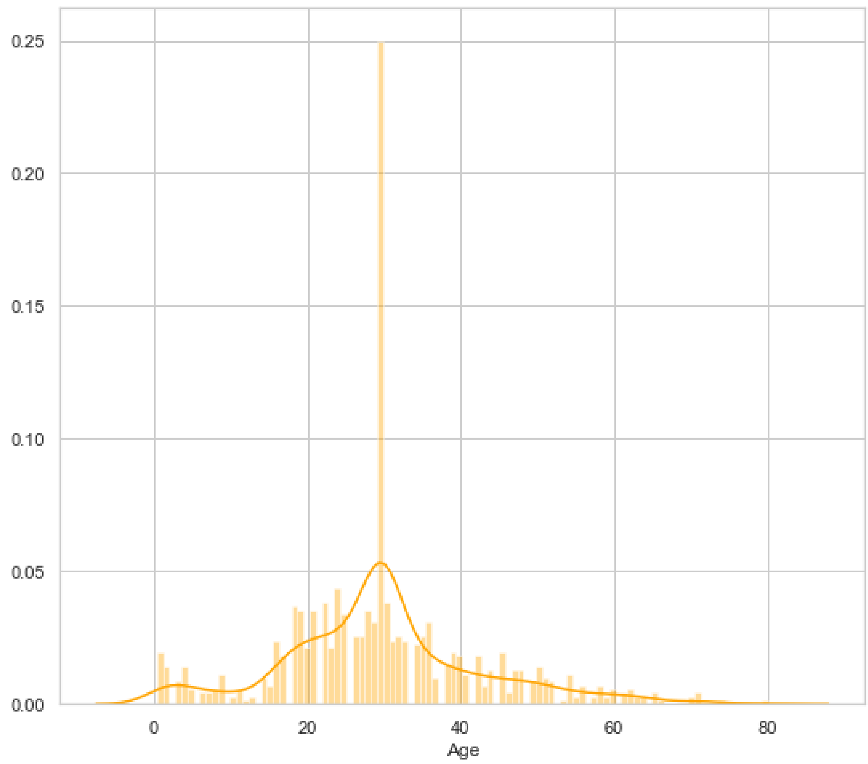
```
# Interactive 3D plot of Pclass vs Survived vs Sex
myData2 = myData[["Pclass","Survived","Sex"]] # Interactive 3D Plot
data = myData2.iplot(kind='surface', colorscale='rdylbu')# Plot the Interactice chart
```

- Interactive 3D plot shows Survival rate by Pclass and sex

Dist plot

In [115]:

```
# Distplot of Age
plt.figure(figsize=(9, 8)) # histograms to plot Age
sns.distplot(myData['Age'], color='orange', bins=100, hist_kws={'alpha': 0.4});# Plot the dist plot
```



- The above dist plot shows that there is a spike of passengers in between age group 25-35

Preparing data for machine learning algorithms

In [478]:

```
import pandas as pd
```

In [116]:

```
df=pd.read_csv('C:\\Users\\15512\\OneDrive\\Desktop\\titanic.csv')# Reading first 5 rows of Datase
t
df
```

Out[116]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	0	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	1	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	1	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	1	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	0	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	0	29.7	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	0	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	0	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	1	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	1	14.0	1	0	237736	30.0708	NaN	C
10	11	1	3	Sandstrom, Miss. Marguerite Rut	1	4.0	1	1	PP 9549	16.7000	G6	S
11	12	1	1	Bonnell, Miss. Elizabeth	1	58.0	0	0	113783	26.5500	C103	S

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
12	13	0	3	Saundercock, Mr. William Henry	0	20.0	0	0	A/5. 2151	8.0500	NaN	S
13	14	0	3	Andersson, Mr. Anders Johan	0	39.0	1	5	347082	31.2750	NaN	S
14	15	0	3	Vestrom, Miss. Hulda Amanda Adolfina	1	14.0	0	0	350406	7.8542	NaN	S
15	16	1	2	Hewlett, Mrs. (Mary D Kingcome)	1	55.0	0	0	248706	16.0000	NaN	S
16	17	0	3	Rice, Master. Eugene	0	2.0	4	1	382652	29.1250	NaN	Q
17	18	1	2	Williams, Mr. Charles Eugene	0	29.7	0	0	244373	13.0000	NaN	S
18	19	0	3	Vander Planke, Mrs. Julius (Emelia Maria Vande...	1	31.0	1	0	345763	18.0000	NaN	S
19	20	1	3	Masselmani, Mrs. Fatima	1	29.7	0	0	2649	7.2250	NaN	C
20	21	0	2	Fynney, Mr. Joseph J	0	35.0	0	0	239865	26.0000	NaN	S
21	22	1	2	Beesley, Mr. Lawrence	0	34.0	0	0	248698	13.0000	D56	S
22	23	1	3	McGowan, Miss. Anna "Annie"	1	15.0	0	0	330923	8.0292	NaN	Q
23	24	1	1	Sloper, Mr. William Thompson	0	28.0	0	0	113788	35.5000	A6	S
24	25	0	3	Palsson, Miss. Torborg Danira	1	8.0	3	1	349909	21.0750	NaN	S
25	26	1	3	Asplund, Mrs. Carl Oscar (Selma Augusta Emilia...	1	38.0	1	5	347077	31.3875	NaN	S
26	27	0	3	Emir, Mr. Farred Chehab	0	29.7	0	0	2631	7.2250	NaN	C
27	28	0	1	Fortune, Mr. Charles Alexander	0	19.0	3	2	19950	263.0000	C23 C25 C27	S
28	29	1	3	O'Dwyer, Miss. Ellen "Nellie"	1	29.7	0	0	330959	7.8792	NaN	Q
29	30	0	3	Todoroff, Mr. Lalio	0	29.7	0	0	349216	7.8958	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
861	862	0	2	Giles, Mr. Frederick Edward	0	21.0	1	0	28134	11.5000	NaN	S
862	863	1	1	Swift, Mrs. Frederick Joel (Margaret Welles Ba...	1	48.0	0	0	17466	25.9292	D17	S
863	864	0	3	Sage, Miss. Dorothy Edith "Dolly"	1	29.7	8	2	CA. 2343	69.5500	NaN	S
864	865	0	2	Gill, Mr. John William	0	24.0	0	0	233866	13.0000	NaN	S
865	866	1	2	Bystrom, Mrs. (Karolina)	1	42.0	0	0	236852	13.0000	NaN	S
866	867	1	2	Duran y More, Miss. Asuncion	1	27.0	1	0	SC/PARIS 2149	13.8583	NaN	C
867	868	0	1	Roebling, Mr. Washington Augustus II	0	31.0	0	0	PC 17590	50.4958	A24	S
868	869	0	3	van Melkebeke, Mr. Philemon	0	29.7	0	0	345777	9.5000	NaN	S
869	870	1	3	Johnson, Master. Harold Theodor	0	4.0	1	1	347742	11.1333	NaN	S
870	871	0	3	Balkic, Mr. Cerin	0	26.0	0	0	349248	7.8958	NaN	S
871	872	1	1	Beckwith, Mrs. Richard Leonard (Sallie Monypeny)	1	47.0	1	1	11751	52.5542	D35	S
872	873	0	1	Carlsson, Mr. Frans Olof	0	33.0	0	0	695	5.0000	B51 B53 B55	S
873	874	0	3	Vander Cruyssen, Mr. Victor	0	47.0	0	0	345765	9.0000	NaN	S
874	875	1	2	Abelson, Mrs. Samuel (Hannah Wizosky)	1	28.0	1	0	P/PP 3381	24.0000	NaN	C
875	876	1	3	Najib, Miss. Adele Kiamie "Jane"	1	15.0	0	0	2667	7.2250	NaN	C
876	877	0	3	Gustafsson, Mr. Alfred Ossian	0	20.0	0	0	7534	9.8458	NaN	S
877	878	0	3	Petroff, Mr. Nedelio	0	19.0	0	0	349212	7.8958	NaN	S
878	879	0	3	Laleff, Mr. Kristo	0	29.7	0	0	349217	7.8958	NaN	S
879	880	1	1	Potter, Mrs. Thomas Jr (Lily Alexenia Wilson)	1	56.0	0	1	11767	83.1583	C50	C
880	881	1	2	Shelley, Mrs. William (Imanita Parrish Hall)	1	25.0	0	1	230433	26.0000	NaN	S
881	882	0	3	Markun, Mr. Johann	0	33.0	0	0	349257	7.8958	NaN	S
882	883	0	3	Dahlberg, Miss. Gerda Ulrika	1	22.0	0	0	7552	10.5167	NaN	S

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
883	884	0	2	Banfield, Mr. Frederick James	0	28.0	0	0	C.A./SOTON 34068	10.5000	NaN	S
884	885	0	3	Sutehall, Mr. Henry Jr	0	25.0	0	0	SOTON/OQ 392076	7.0500	NaN	S
885	886	0	3	Rice, Mrs. William (Margaret Norton)	1	39.0	0	5	382652	29.1250	NaN	Q
886	887	0	2	Montvila, Rev. Juozas	0	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	1	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	1	29.7	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	0	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	0	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 12 columns

In [117]:

```
df.drop(["Name","Ticket","Cabin",], axis=1, inplace=True)
df.head() # Removing unnecessary rows and columns
```

Out[117]:

	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	1	0	3	0	22.0	1	0	7.2500	S
1	2	1	1	1	38.0	1	0	71.2833	C
2	3	1	3	1	26.0	0	0	7.9250	S
3	4	1	1	1	35.0	1	0	53.1000	S
4	5	0	3	0	35.0	0	0	8.0500	S

One Hot Encoding for Categorical variables

In [118]:

```
df_onehot = df.copy()# Copy of data
df_onehot = pd.get_dummies(df_onehot,columns=['Sex','Embarked'],prefix =['Sex','Embarked'])#
Getting dummies
print(df_onehot.head())# First five rows after one hot encoding
```

```
   PassengerId  Survived  Pclass   Age  SibSp  Parch     Fare  Sex_0  Sex_1  \
0            1         0       3  22.0      1      0   7.2500      1      0
1            2         1       1  38.0      1      0  71.2833      0      1
2            3         1       3  26.0      0      0   7.9250      0      1
3            4         1       1  35.0      1      0  53.1000      0      1
4            5         0       3  35.0      0      0   8.0500      1      0

   Embarked_C  Embarked_Q  Embarked_S
0           0           0           1
1           1           0           0
2           0           0           1
3           0           0           1
4           0           0           1
```

In [119]:

```
X = df_onehot.drop(['Survived'],axis=1)  # definde X and y
y = df_onehot.Survived
```

In [120]:

```
# splitting the datset into Training set and test set
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size =0.2 ,random_state=0)# Split the dat
a set
```

```
a set
```

Splitting the dats set

In [121]:

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size =0.2 , random_state = 0)# Split the da
ta set
```

Multiple Linear Regression Model

In [123]:

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)# Training the model
```

Out[123]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [124]:

```
import numpy as np
```

In [125]:

```
y_pred = regressor.predict(X_test)# predict the test set results
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1))))# print y prediction
print(y_test) # print y test
```

```
[ 0.15  0.1   0.14  0.98  0.64  0.44  0.91  0.95  0.49  0.65  0.08  0.71
  0.18  0.9   1.04  0.71  0.15  0.29  0.07  0.32  0.34  1.03  0.18  0.44
  0.66  0.89  0.09  0.66  0.8   0.63  0.13  0.64  0.11  0.44  0.06  0.47
  0.02  0.28  0.3   0.11  0.25  0.18  0.1   0.01  0.88  0.1   0.1   1.02
  0.21  0.25  0.44  0.53  0.88  0.17  0.48  0.23  0.23  0.52  0.07  0.09
  0.2   0.57  0.8   0.49  0.6   0.17  0.82  0.27  0.9   1.01  0.77  0.28
  0.46  0.1   0.14  0.63  0.41  0.42  0.07  0.32  0.14  0.18  0.74  0.19
  0.17  1.01  0.97  0.39  0.74  0.54  0.43  0.17  0.39  0.89  0.58  0.15
  0.79  0.    0.24  0.51 -0.04  0.07  0.17  0.12  0.55  0.52  0.78  0.49
  0.26  0.64 -0.02  0.99  0.1   0.61  0.36  0.78  0.6   1.07  0.04  0.62
  0.12  0.18  0.16  0.38  0.08  0.3   0.15  0.09  0.27  0.13  0.74  0.16
  0.13  0.62  0.22  0.16  0.14  0.49  0.13  0.31  0.28  0.97  0.1   0.76
  0.78  0.64  0.27  0.65  0.91  0.1   0.31  0.61  0.56  0.11  0.9   0.29
  0.54  0.06  0.69  0.74  0.1   0.21  0.79  0.45  0.18  0.13 -0.04  0.15
  0.14  0.18 -0.03  0.9   0.1   0.1   0.76  0.1   0.96  0.14  0.17]
495    0
648    0
278    0
31     1
255    1
298    1
609    1
318    1
484    1
367    1
704    0
346    1
196    0
535    1
310    1
14     0
350    0
145    0
614    0
803    1
144    0
708    1
778    0
```

```
270    0
474    0
319    1
519    0
141    1
880    1
642    0
      ..
158    0
62     0
79     1
503    0
231    0
389    1
619    0
362    0
570    1
264    0
644    1
384    0
762    1
513    1
85     1
352    0
75     0
631    0
395    0
294    0
500    0
222    0
1      1
425    0
760    0
780    1
837    0
215    1
833    0
372    0
Name: Survived, Length: 179, dtype: int64
```

Evaluating Multiple Linear Regression Model

In [126]:

```
results = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})# actual vs predicted
results # print the results
```

Out[126]:

	Actual	Predicted
495	0	0.149220
648	0	0.101651
278	0	0.136280
31	1	0.975928
255	1	0.637299
298	1	0.439623
609	1	0.911846
318	1	0.952392
484	1	0.485169
367	1	0.651496
704	0	0.078633
346	1	0.707056
196	0	0.182104
535	1	0.901224
310	1	1.040536

	Actual	Predicted
14	0	0.708202
350	0	0.145908
145	0	0.287882
614	0	0.067812
803	1	0.324337
144	0	0.344475
708	1	1.026748
778	0	0.180700
270	0	0.439833
474	0	0.656246
319	1	0.894974
519	0	0.087278
141	1	0.656450
880	1	0.795264
642	0	0.628038
...	...	...
158	0	0.103148
62	0	0.310244
79	1	0.606533
503	0	0.559699
231	0	0.107218
389	1	0.899716
619	0	0.291630
362	0	0.544611
570	1	0.060374
264	0	0.685974
644	1	0.736202
384	0	0.102385
762	1	0.208854
513	1	0.793697
85	1	0.447952
352	0	0.184598
75	0	0.133265
631	0	-0.035346
395	0	0.151818
294	0	0.139236
500	0	0.183949
222	0	-0.034077
1	1	0.901157
425	0	0.102102
760	0	0.103368
780	1	0.757834
837	0	0.101339
215	1	0.957631
833	0	0.144354
372	0	0.171227

179 rows × 2 columns

In [127]:

```
from sklearn.metrics import r2_score# from sklearn import r2_score
```

```
from sklearn.metrics import r2_score# from sklearn import r2 score
r2_score(y_test, y_pred)# r2 score for linear regression model
```

Out[127]:

```
0.42576686014450804
```

* R2 score for multiple linear regression is 0.42

Random Forest Regression Model

In [128]:

```
from sklearn.ensemble import RandomForestRegressor# From sklearn import Random forest regressor
regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
regressor.fit(X_train, y_train)# Train the model
```

Out[128]:

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=10,
                      n_jobs=None, oob_score=False, random_state=0, verbose=0,
                      warm_start=False)
```

In [129]:

```
y_pred = regressor.predict(X_test)# test the model
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1))))# print y pred
print(y_test)# print y test
```

```
[0.6 0.  0.2 0.6 0.1 0.  0.9 0.6 0.3 0.6 0.1 0.8 0.  1.  1.  0.8 0.  0.
 0.  1.  0.2 0.5 0.  0.2 0.5 1.  0.4 0.8 0.9 0.4 0.2 0.9 0.1 0.2 0.  0.8
 0.1 0.  0.  0.2 0.3 0.1 0.1 0.2 1.  0.  0.  0.9 0.  0.2 0.  0.1 1.  0.
 0.1 0.3 0.3 0.  0.  0.  0.3 0.7 1.  0.1 0.6 0.4 0.9 0.1 0.4 1.  1.  0.4
 0.3 0.1 0.  0.9 0.4 0.5 0.  0.3 0.  0.6 1.  0.  0.3 1.  1.  0.3 1.  0.6
 0.1 0.  1.  1.  0.  0.  0.7 0.1 0.8 0.9 0.2 0.  0.  0.2 0.2 0.6 1.  0.1
 0.1 0.4 0.3 1.  0.2 0.1 0.2 1.  0.5 1.  0.2 0.8 0.4 0.4 0.1 0.  0.  0.2
 0.1 0.1 0.  0.  0.9 0.1 0.  0.4 0.3 0.1 0.  0.3 0.1 0.1 0.  1.  0.2 0.7
 1.  0.5 0.2 0.7 1.  0.2 0.2 0.1 0.5 0.3 1.  0.  0.3 0.2 1.  0.8 0.  0.2
 1.  0.2 0.1 0.7 0.  0.6 0.  0.  0.2 0.9 0.  0.  1.  0.  1.  0.  0.6]
495    0
648    0
278    0
31     1
255    1
298    1
609    1
318    1
484    1
367    1
704    0
346    1
196    0
535    1
310    1
14     0
350    0
145    0
614    0
803    1
144    0
708    1
778    0
270    0
474    0
319    1
519    0
```

```
141    1
880    1
642    0
      ..
158    0
62     0
79     1
503    0
231    0
389    1
619    0
362    0
570    1
264    0
644    1
384    0
762    1
513    1
85     1
352    0
75     0
631    0
395    0
294    0
500    0
222    0
1      1
425    0
760    0
780    1
837    0
215    1
833    0
372    0
Name: Survived, Length: 179, dtype: int64
```

Evaluating Random Forest Regression model

In [130]:

```
results = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})# actual vs predicated result
results# Print the results
```

Out[130]:

	Actual	Predicted
495	0	0.6
648	0	0.0
278	0	0.2
31	1	0.6
255	1	0.1
298	1	0.0
609	1	0.9
318	1	0.6
484	1	0.3
367	1	0.6
704	0	0.1
346	1	0.8
196	0	0.0
535	1	1.0
310	1	1.0
14	0	0.8
350	0	0.0
145	0	0.0

	Actual	Predicted
614	0	0.0
803	1	1.0
144	0	0.2
708	1	0.5
778	0	0.0
270	0	0.2
474	0	0.5
319	1	1.0
519	0	0.4
141	1	0.8
880	1	0.9
642	0	0.4
...	...	...
158	0	0.2
62	0	0.2
79	1	0.1
503	0	0.5
231	0	0.3
389	1	1.0
619	0	0.0
362	0	0.3
570	1	0.2
264	0	1.0
644	1	0.8
384	0	0.0
762	1	0.2
513	1	1.0
85	1	0.2
352	0	0.1
75	0	0.7
631	0	0.0
395	0	0.6
294	0	0.0
500	0	0.0
222	0	0.2
1	1	0.9
425	0	0.0
760	0	0.0
780	1	1.0
837	0	0.0
215	1	1.0
833	0	0.0
372	0	0.6

179 rows × 2 columns

In [131]:

```
r2_score(y_test, y_pred)# r2 score for forest regression model
```

Out[131]:

```
0.3738537549407115
```

***R2 score for Random Forest regression is 0.38**

Decision Tree Regression Model

from sklearn.tree import DecisionTreeRegressor # from sklearn Import Decision Tree Regression model

In [132]:

```
regressor = DecisionTreeRegressor(random_state = 0)
regressor.fit(X_train, y_train)# train the model
```

Out[132]:

```
DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=0, splitter='best')
```

In [133]:

```
y_pred = regressor.predict(X_test)# test the model
np.set_printoptions(precision=2)
print(np.concatenate((y_pred.reshape(len(y_pred),1))))
print(y_test)# Print the results of y test
```

```
[0. 0. 0. 1. 0. 0. 1. 1. 1. 1. 0. 1. 0. 1. 1. 1. 0. 0. 0. 1. 0. 0. 0. 0.
 0. 1. 1. 1. 1. 1. 1. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 1.
 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 1. 0. 1. 0. 0. 1. 1. 0.
 0. 0. 0. 1. 1. 0. 0. 0. 0. 1. 1. 0. 0. 1. 1. 0. 1. 1. 0. 0. 1. 1. 0. 0.
 1. 0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 0. 1. 1. 1. 0. 0.
 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1.
 1. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 0.
 0. 0. 0. 1. 0. 0. 1. 0. 1. 0. 1.]
495    0
648    0
278    0
31     1
255    1
298    1
609    1
318    1
484    1
367    1
704    0
346    1
196    0
535    1
310    1
14     0
350    0
145    0
614    0
803    1
144    0
708    1
778    0
270    0
474    0
319    1
519    0
141    1
880    1
642    0
      ..
158    0
62     0
79     1
503    0
```

```
231    0
389    1
619    0
362    0
570    1
264    0
644    1
384    0
762    1
513    1
85     1
352    0
75     0
631    0
395    0
294    0
500    0
222    0
1      1
425    0
760    0
780    1
837    0
215    1
833    0
372    0
Name: Survived, Length: 179, dtype: int64
```

Evaluating Decision Tree Regression Model

In [134]:

```
results = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})# actual vs predicated
results# show the result
```

Out[134]:

	Actual	Predicted
495	0	0.0
648	0	0.0
278	0	0.0
31	1	1.0
255	1	0.0
298	1	0.0
609	1	1.0
318	1	1.0
484	1	1.0
367	1	1.0
704	0	0.0
346	1	1.0
196	0	0.0
535	1	1.0
310	1	1.0
14	0	1.0
350	0	0.0
145	0	0.0
614	0	0.0
803	1	1.0
144	0	0.0
708	1	0.0
778	0	0.0

	Actual	Predicted
270	0	0.0
474	0	0.0
319	1	1.0
519	0	1.0
141	1	1.0
880	1	1.0
642	0	1.0
...	...	...
158	0	0.0
62	0	0.0
79	1	0.0
503	0	0.0
231	0	0.0
389	1	1.0
619	0	0.0
362	0	0.0
570	1	0.0
264	0	1.0
644	1	0.0
384	0	0.0
762	1	0.0
513	1	1.0
85	1	0.0
352	0	0.0
75	0	0.0
631	0	0.0
395	0	0.0
294	0	0.0
500	0	0.0
222	0	0.0
1	1	1.0
425	0	0.0
760	0	0.0
780	1	1.0
837	0	0.0
215	1	1.0
833	0	0.0
372	0	1.0

179 rows × 2 columns

In [135]:

```
r2_score(y_test, y_pred)# r2 score for decision tree
```

Out[135]:

```
0.10382081686429523
```

***R2 score for decision tree regression is 0.1274**

Kmeans Clustering

Kmeans Clustering

In [136]:

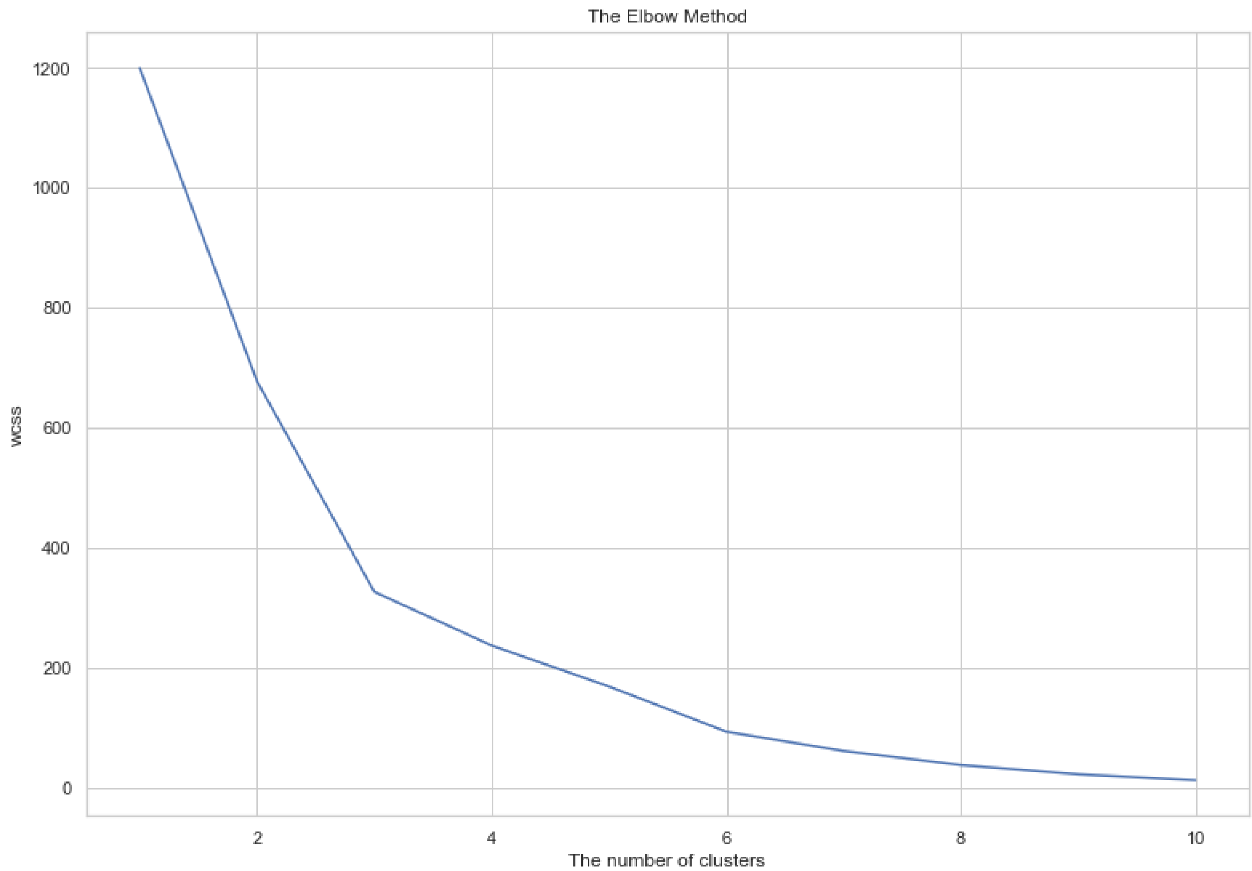
```
x = myData.iloc[:,[2,6]]   # definde X and y
```

In [137]:

```
from sklearn.cluster import KMeans # kmeans clustering
```

In [138]:

```
wcss = [] # intialize empty list
for i in range (1,11): # number of clusters need to be adentifie
    kmeans = KMeans(n_clusters = i , init = 'k-means++',random_state= 42) # first pramter = numb of
cluster
                                                                                    # Init = intializing the
means by kmeans ++
                                                                                    # random_state to find tl
exact numbers
    kmeans.fit(x)  # train the kmeans by Latitude Longitude
    wcss.append(kmeans.inertia_) # to get wcss we need to append new values which to call object
plt.plot(range(1,11),wcss) # plot the number of clusters
plt.title ("The Elbow Method") # plot the title
plt.xlabel("The number of clusters") # plot the x label
plt.ylabel("wcss") # plot the y label
plt.show() # graph elbow method
```



In [139]:

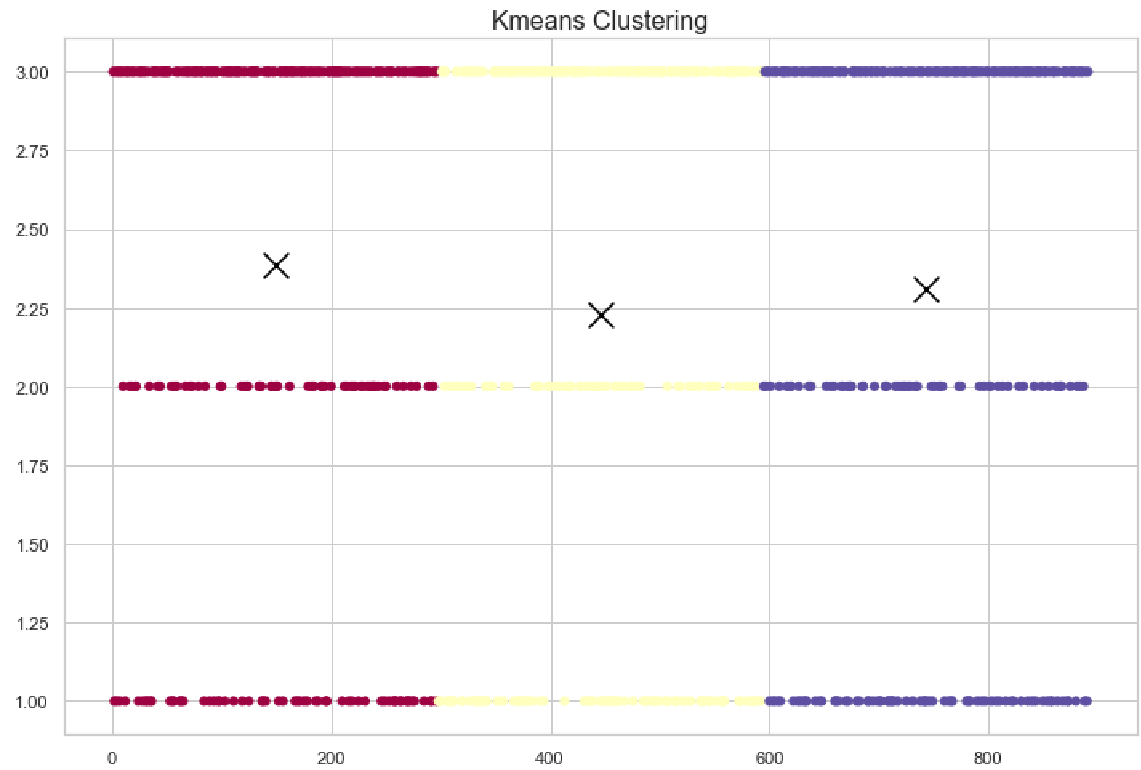
```
kmeans = KMeans(n_clusters = 3 , init = 'k-means++',random_state= 42)
y_kmeans = kmeans.fit_predict(X)  # train and predict the models
print(y_kmeans) # it shows every groups on clusters 2
```

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2]
```

In [140]:

```
fig = plt.figure(figsize=(12,8)) # figure size
plt.scatter(X.values[:,0], X.values[:,1], c=kmeans.labels_, cmap="Spectral", s=25) # plot the scatt
er
plt.scatter(kmeans.cluster_centers_[:,0] ,kmeans.cluster_centers_[:,1], color='black', marker="x",
s=250)
plt.title("Kmeans Clustering", fontsize=16) #plot the title
plt.show() # plot the graph
```



The elbow method is a technique to choose the most number of clusters. The graph above shows that there are 3 clusters in the data set.

Conclusion :

- Survival ratio is high in females than males
- More number of passemgers are in the age group 25-25
- More number of passengers are from point of embarkation southhampston
- More number of passengers have surbived in Pclass1
- More number of passengers travelled in Pclass 3
- More number of male passengers are than the female passengers

- More number of male passengers are than the female passengers
- Survival rate is more in Southhampston embarkment than cherbourg and queenstown
- Age group who have survived more are in the range of 25-30
- Youngest traveller is below 1 yr and oldest traveller is 80 yr old
- Out of total 891 passengers 342 have survived and 549 have died
- Avg age of passengers is 30.Avg male age is 24 and Avg fem age is 23
- Females survival rate is 74% and males survival rate is 19%
- R2 score for multiple linear regression is 0.42 which is best score to test new data for predicting survival rate.

*End of Project **

In [ ]:

```

```