1 Importing the Libraries

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style="whitegrid")
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
# ML Libraries:
from sklearn.preprocessing import OneHotEncoder
from sklearn.linear_model import LogisticRegression
```

In [3]:
```
import plotly.figure_factory as ff
import plotly.offline as py
import plotly.graph_objs as go
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot

from plotly import tools
py.init_notebook_mode (connected = True)

import cufflinks as cf
cf.go_offline()
```

2 Importing the Dataset and General View

In [5]:
```
dftrain = pd.read_csv("train.csv")
dftest = pd.read_csv("test.csv")
```

Lets Concatenate both the data frames for Exploratory Data Analysis :

In [6]:
```
df = pd.concat([dftrain, dftest], axis = 0 )
```

In [7]:
```
# Head of Training set:
dftrain.head(5)
```

Out[7]:

	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 [ ]:
```
```

In [8]:
```
# Breif information about the Data Sets
dftrain.info()
print("_____________________________________")
dftest.info()
```

```
Embarked      889 non-null object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.6+ KB
_____________________________________
<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 [9]:
```
# General Description about the Training Data set:
dftrain.drop(["PassengerId"], axis =1).describe()
```

Out[9]:

	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

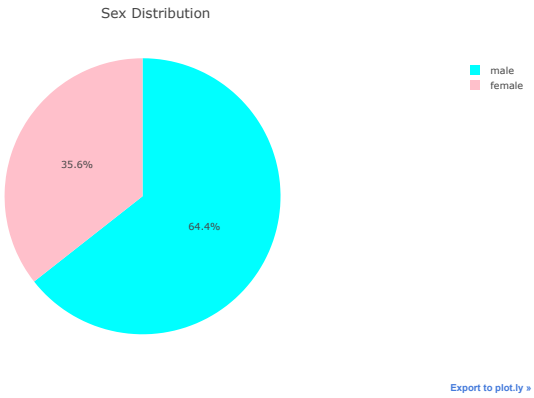
3. Data Visualisation of Entire Data Frame (Training and Test Set)

In [10]:
```
classd= {1:"First Class", 2: "Second Class", 3: "Third Class"}
df["Class"] = df["Pclass"].map(classd)

first = df[df["Class"]=="First Class"]
sec = df[df["Class"]=="Second Class"]
thrd= df[df["Class"]=="Third Class"]

male= df[df["Sex"]=="male"]
female= df[df["Sex"]=="female"]
```

3. a. SEX

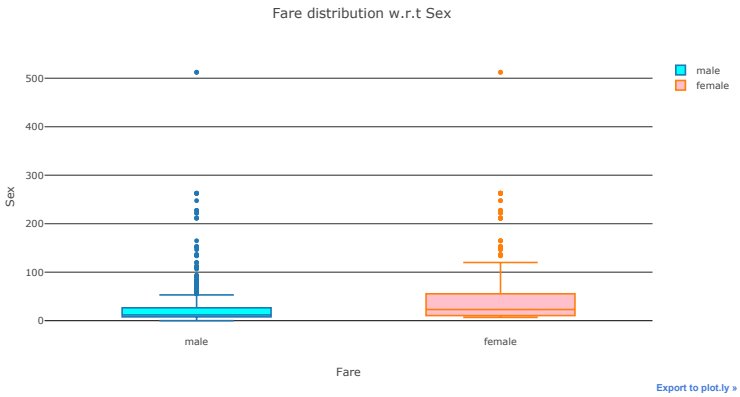
In [11]:
```
#1. Pie Chart for Sex count
sex_count = df["Sex"].value_counts()
print(sex_count)

colors = ['aqua', 'pink']

trace= go.Pie(labels = sex_count.index,
              values = sex_count.values, marker=dict(colors=colors))

layout = go.Layout(title = "Sex Distribution")

data = [trace]
fig = go.Figure(data=data, layout=layout)
py.iplot(fig)
```

```
male      843
female    466
Name: Sex, dtype: int64
```

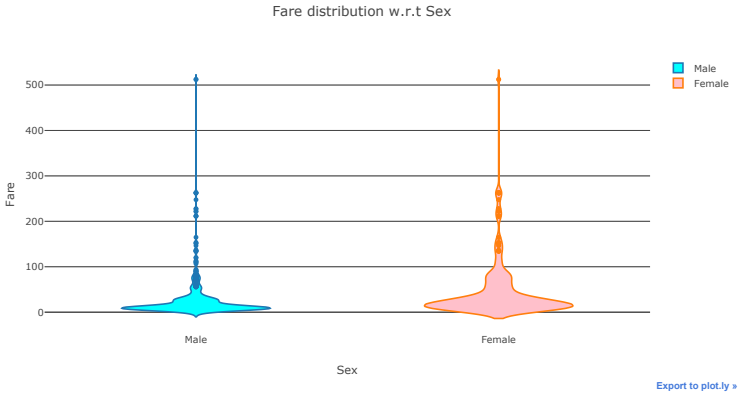


Export to plot.ly »

In [12]:
```
# 1 Boxplot
trace = go.Box(y = male["Fare"],fillcolor="aqua", name= "male" )
trace1 = go.Box(y = female["Fare"], fillcolor="pink", name= "female" )

layout = go.Layout(title="Fare distribution w.r.t Sex", yaxis=dict(title="Sex"), xaxis= dict(title="Fare"))

data=[trace, trace1]
fig = go.Figure(data = data, layout=layout)
py.iplot(fig)

# 2 Violin Plot
trace1 = go.Violin( y = male["Fare"], fillcolor="aqua", name="Male")
trace2 = go.Violin( y = female["Fare"],fillcolor="pink", name="Female")

layout = go.Layout(title="Fare distribution w.r.t Sex", yaxis=dict(title="Fare"), xaxis= dict(title="Sex"))

data=[trace1, trace2]
fig = go.Figure(data = data, layout=layout)
py.iplot(fig)
```

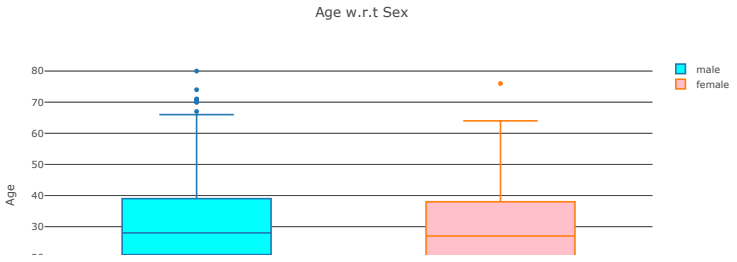


Export to plot.ly »



Export to plot.ly »

In [13]:
```
#male= df[df["Sex"]=="male"]
#female= df[df["Sex"]=="female"]

# Box
trace = go.Box(y = male["Age"],fillcolor="aqua", name= "male" )
tracel = go.Box(y = female["Age"], fillcolor="pink", name= "female" )
layout = go.Layout(title="Age w.r.t Sex", yaxis=dict(title="Age"), xaxis= dict(title="Sex"))

data=[trace, tracel]
fig = go.Figure(data = data, layout=layout)
py.iplot(fig)
```

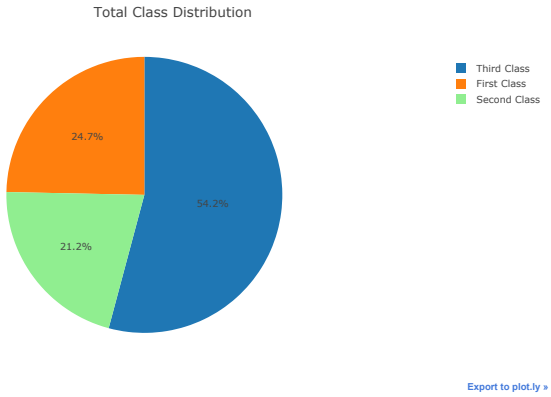


In [ ]:
```
```

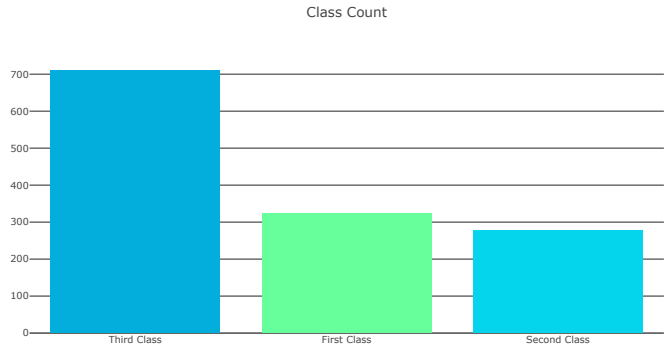
2.b. Class

In [14]:
```
# Pie Chart
class_count = df["Class"].value_counts()
print(class_count)

colors = ["lightorange", "lightpurple", "lightgreen"]
trace = go.Pie( labels= class_count.index,values = class_count.values, marker = dict(colors = colors))

layout = go.Layout( title = "Total Class Distribution")

data= [trace]
fig = go.Figure(data= data, layout= layout)
py.iplot(fig)
```

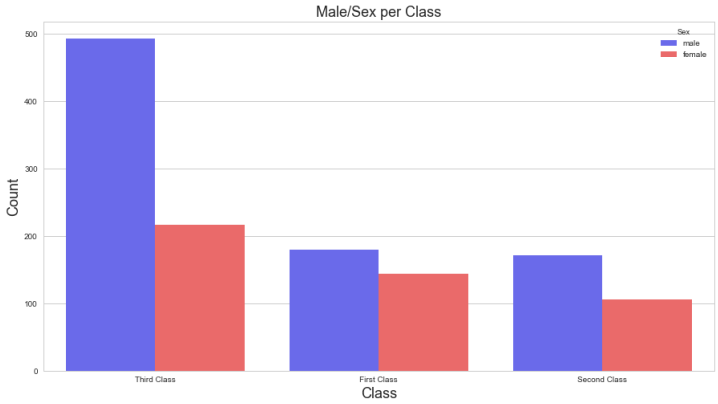
```
Third Class     709
First Class     323
Second Class    277
Name: Class, dtype: int64
```



In [15]:
```
# classd= {1:"First Class", 2: "Second Class", 3: "Third Class"}
#df["Class"] = df["Pclass"].map(classd)

# Bar Plot
trace1 = go.Bar(x=class_count.index, y= class_count.values,
                marker=dict(
                color=dftrain["Age"],
                colorscale = 'Jet'))

layout1 = go.Layout(title = "Class Count" )

data= [trace1]
fig = go.Figure(data= data, layout= layout1)
py.iplot(fig)


#plt plot
plt.figure(figsize=(15,8))
sns.countplot(x= df["Class"], hue= df["Sex"], palette="seismic")
plt.title("Male/Sex per Class", fontsize = 18)
plt.xlabel("Class", fontsize = 18)
plt.ylabel("Count", fontsize = 18)
plt.show()
```

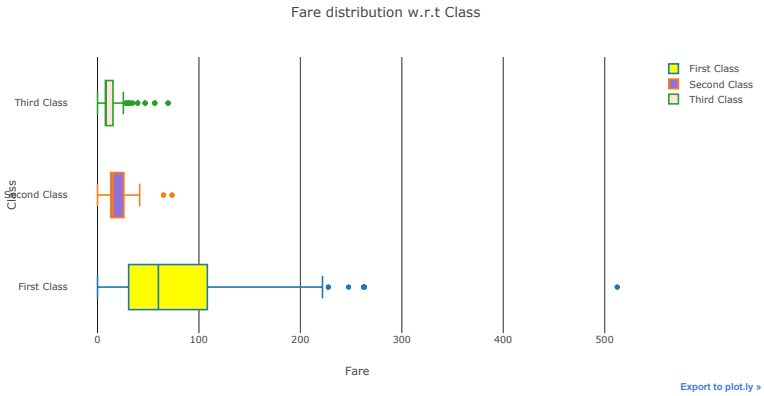


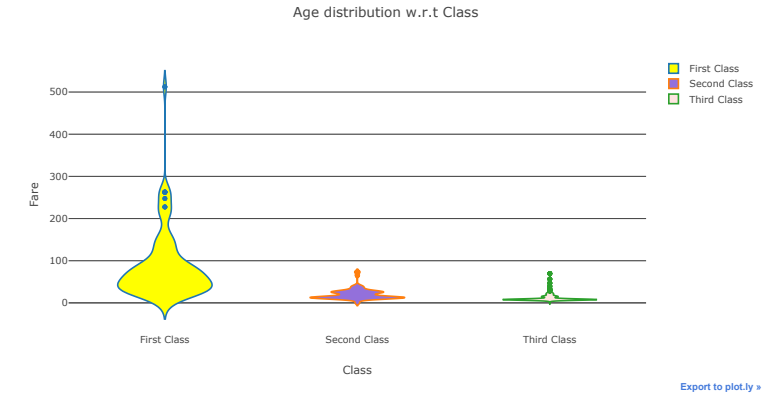
Export to plot.ly »



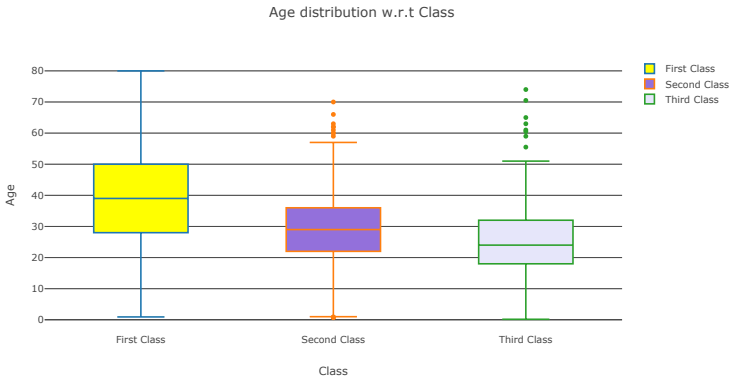
In [16]:

```
#first = dftrain[dftrain["Class"]=="First Class"]
#sec = dftrain[dftrain["Class"]=="Second Class"]
#thrd= dftrain[dftrain["Class"]=="Third Class"]

# Box plot
trace1 = go.Box( x = first["Fare"], fillcolor="yellow", name="First Class")
trace2 = go.Box( x = sec["Fare"],fillcolor="mediumpurple", name="Second Class")
trace3 = go.Box( x = thrd["Fare"], fillcolor="mistyrose", name="Third Class")

layout = go.Layout(title="Fare distribution w.r.t Class", yaxis=dict(title="Class"), xaxis= dict(title="Fare"))

data=[trace1, trace2, trace3]
fig = go.Figure(data = data, layout=layout)
py.iplot(fig)

# Violin Plot
trace1 = go.Violin( y = first["Fare"], fillcolor="yellow", name="First Class")
trace2 = go.Violin( y = sec["Fare"],fillcolor="mediumpurple", name="Second Class")
trace3 = go.Violin( y = thrd["Fare"], fillcolor="mistyrose", name="Third Class")

layout = go.Layout(title="Age distribution w.r.t Class", yaxis=dict(title="Fare"), xaxis= dict(title="Class"))

data=[trace1, trace2, trace3]
fig = go.Figure(data = data, layout=layout)
py.iplot(fig)

# violinplot
#plt.figure(figsize=(14,5))
#sns.violinplot(x = df["Class"], y=df["Age"], palette="magma")
#plt.xlabel("Class", fontsize =20)
#plt.ylabel("Age", fontsize =20)
#plt.title("Violin plot of Class w.r.t Age",fontsize =20)
#plt.show()
```



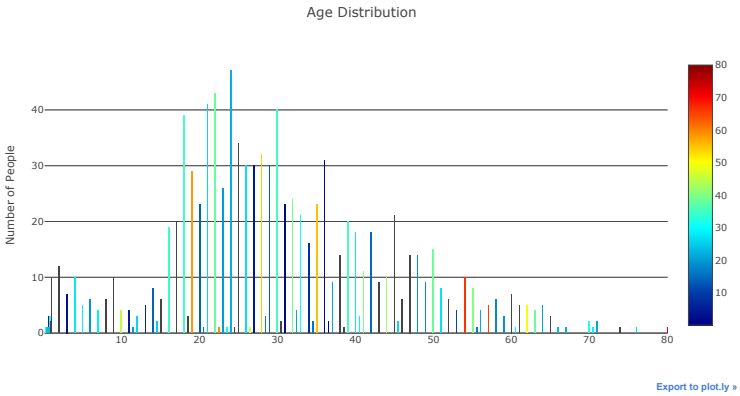


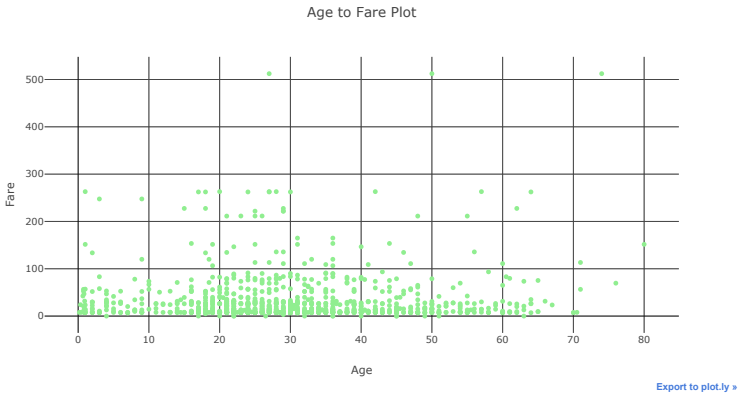
In [17]:
```
# Box Plot Age Vs Class
# Box plot
trace1 = go.Box( y = first["Age"], fillcolor="yellow", name="First Class")
trace2 = go.Box( y = sec["Age"],fillcolor="mediumpurple", name="Second Class")
trace3 = go.Box( y = thrd["Age"], fillcolor="lavender", name="Third Class")

layout = go.Layout(title="Age distribution w.r.t Class", yaxis=dict(title="Age"), xaxis= dict(title="Class"))

data=[trace1, trace2, trace3]
fig = go.Figure(data = data, layout=layout)
py.iplot(fig)
```



2.c. Age Distribution

In [18]:
```
# Age Bar plot
age_count = df["Age"].dropna().value_counts()

trace = go.Bar(x = age_count.index,
               y = age_count.values,
               marker = dict(color = df["Age"],
                             colorscale = "Jet",
                             showscale = True))
layout = go.Layout(title = "Age Distribution",
                   yaxis = dict(title = "Number of People"))
data = [trace]
fig = go.Figure(data = data, layout = layout)
py.iplot(fig)


# Age Vs Fare
trace=go.Scatter(x = df["Age"].dropna(), y=df["Fare"], mode = "markers",
                 marker = dict(size = 6, color = "lightgreen"))

layout = go.Layout(title ="Age to Fare Plot", xaxis= dict(title = "Age"),
                   yaxis= dict(title = "Fare"))

data = [trace]
fig = go.Figure(data=data, layout=layout)
py.iplot(fig)
```

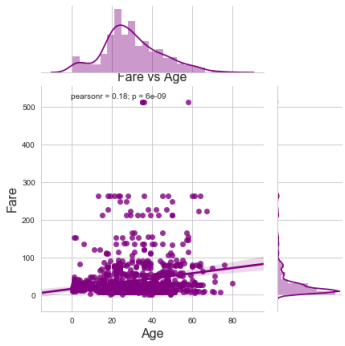




2.d. Joint plot: Distribution w.r.t to AGE & FARE

In [20]:
```
plt.figure(figsize=(14,5))
sns.jointplot(x=df["Age"], y=df["Fare"], kind="reg",
              color="purple", ratio=3,dropna= True)
plt.xlabel("Age", fontsize =16)
plt.ylabel("Fare", fontsize =16)
plt.title("Fare vs Age",fontsize =16)
plt.show()
```

```
<matplotlib.figure.Figure at 0x1a217df518>
```

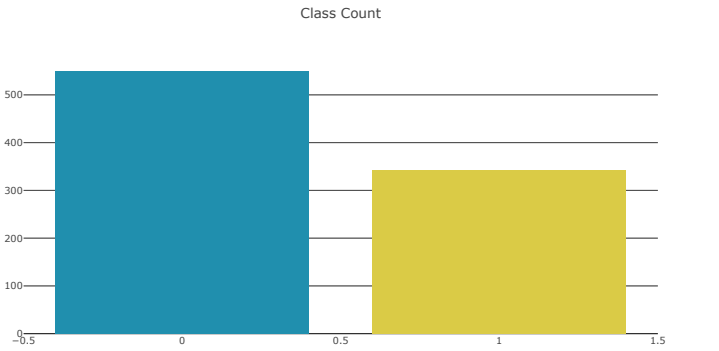


2.e. Survived Plots

In [21]:
```
df["Survived"].value_counts()
```

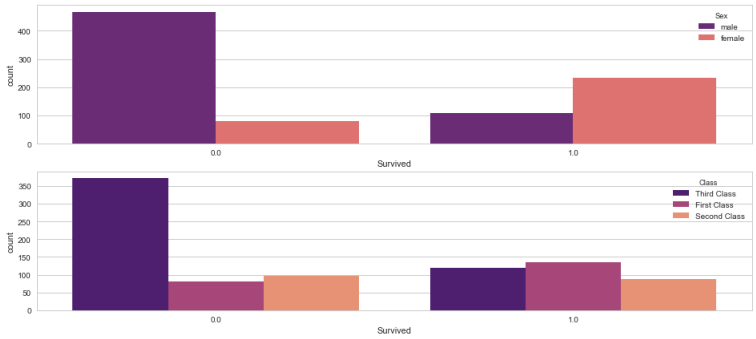
Out[21]:
```
0.0    549
1.0    342
Name: Survived, dtype: int64
```

In [22]:
```
# Survival count
sur_count = df["Survived"].value_counts()
trace1 = go.Bar(x=sur_count.index, y= sur_count.values,
                marker=dict(
                color=df["Age"],
                colorscale = 'Portland'))

layout1 = go.Layout(title = "Class Count" )

data= [trace1]
fig = go.Figure(data= data, layout= layout1)
py.iplot(fig)
```



Looks like 1098 people did not survive, while around 684 people in both the dataset survived

In [23]:
```
# Survival with hue of Sex

fig, (ax1, ax2) = plt.subplots(2, figsize= (16,7))
sns.countplot(x="Survived", hue="Sex",  data=df, palette="magma", ax=ax1)
sns.countplot(x="Survived", hue="Class",  data=df, palette="magma", ax=ax2)

plt.show()
plt.tight_layout()
```




```
<matplotlib.figure.Figure at 0x1a18830b70>
```

We can see a trend here. It looks like people who couldn't survived were much more like to be male. While on the other hand people who survived are more likely to be female.

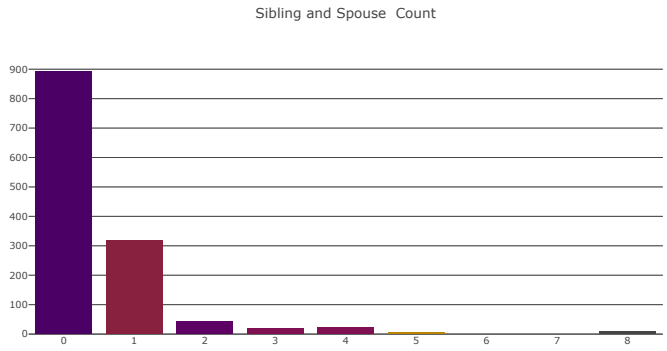
In [24]:
```
plt.figure(figsize= (16,7))
sns.scatterplot(data=df, x="Age", y="Fare", hue="Sex")
plt.show()
plt.figure(figsize= (16,7))
sns.scatterplot(data=df, x="Age", y="Fare", hue="Survived")
plt.show()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-24-18e7e870d26f> in <module>()
      1 plt.figure(figsize= (16,7))
----> 2 sns.scatterplot(data=df, x="Age", y="Fare", hue="Sex")
      3 plt.show()
      4 plt.figure(figsize= (16,7))
      5 sns.scatterplot(data=df, x="Age", y="Fare", hue="Survived")

AttributeError: module 'seaborn' has no attribute 'scatterplot'


<matplotlib.figure.Figure at 0x1a18830358>
```

In [ ]:

2.f. Sibling and Spouse on board

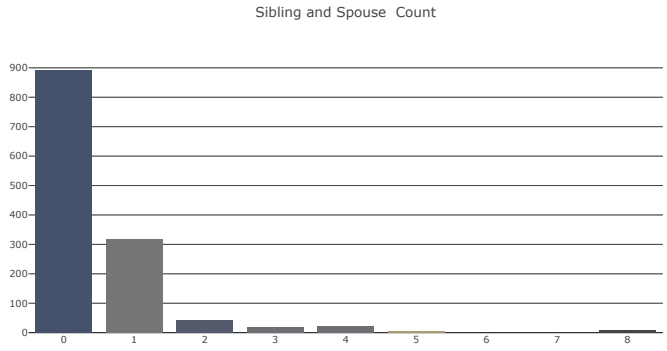
In [25]:
```
sib_count = df["SibSp"].value_counts()
trace1 = go.Bar(x=sib_count.index, y= sib_count.values,
                marker=dict(
                color=df["Age"],
                colorscale = 'Electric'))

layout1 = go.Layout(title = "Sibling and Spouse  Count" )

data= [trace1]
fig = go.Figure(data= data, layout= layout1)
py.iplot(fig)
```



Export to plot.ly »

2. g. Parent and child on board

In [26]:
```
# Count plot for number of sibling or Spouse aboard
par_count = df["SibSp"].value_counts()
trace1 = go.Bar(x=par_count.index, y= par_count.values,
                marker=dict(
                color=df["Age"],
                colorscale = 'Cividis'))

layout1 = go.Layout(title = "Sibling and Spouse  Count" )

data= [trace1]
fig = go.Figure(data= data, layout= layout1)
py.iplot(fig)
```

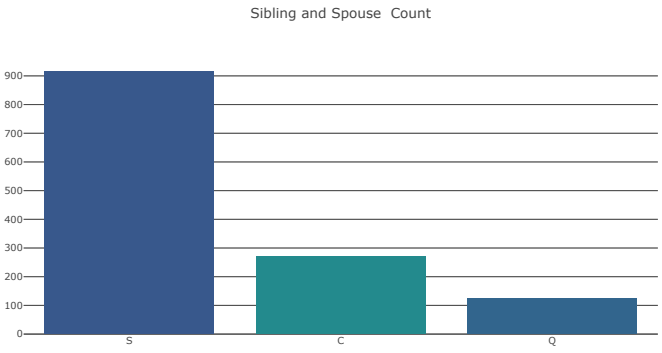


Export to plot.ly »

In [ ]:

3. g. Embarked Ship

In [27]:
```
em_count = df["Embarked"].value_counts()
trace1 = go.Bar(x=em_count.index, y= em_count.values,
                marker=dict(
                color=df["Age"],
                colorscale = 'Viridis'))

layout1 = go.Layout(title = "Sibling and Spouse  Count" )

data= [trace1]
fig = go.Figure(data= data, layout= layout1)
py.iplot(fig)
```

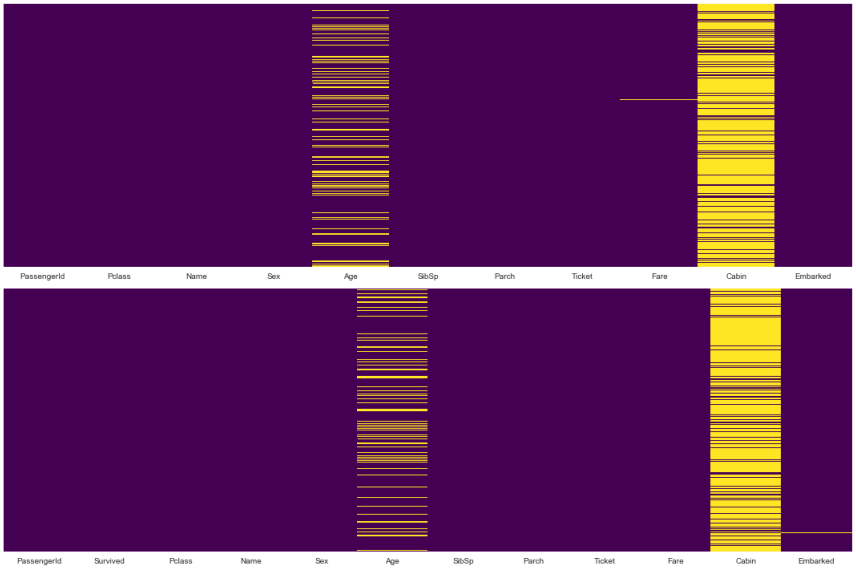


Export to plot.ly »

In [ ]:
```
```

4 Data Visualisation for Data Preprocessing

In [28]:
```
fig, (ax1,ax2) = plt.subplots (2, figsize = (15,10))
sns.heatmap(dftest.isnull(), cmap="viridis", yticklabels=False
            ,cbar = False, ax= ax1)
sns.heatmap(dftrain.isnull(), cmap="viridis", yticklabels=False
            ,cbar = False, ax= ax2)
plt.tight_layout()
```



So we can check out that we're missing some age information and we are missing a lot of Cabin information. Roughly about 20 percent of that age data is missing and the proportion of age missing is likely small enough for a reasonable replacement of some form of imputation meaning I can actually use the knowledge of the other columns to fill in reasonable values for that age column. Looking at the cabin column however it looks like we're just missing too much of that data to do something useful with it at a basic level. We're going to go ahead and probably drop this later or change it to send up some other feature like

5 Data Preprocessing

5.a. Dealing with MISSING VALUES:

In [29]:
```
# on the basis of the box plot of age vs class.
# We can impute the average age on the basis of that plot
def imput_age(col):
    Age = col[0]
    Pclass = col[1]

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

dftrain["Age"] = dftrain[["Age", "Pclass"]].apply (imput_age, axis=1)
dftest["Age"] = dftest[["Age", "Pclass"]].apply (imput_age, axis=1)
```

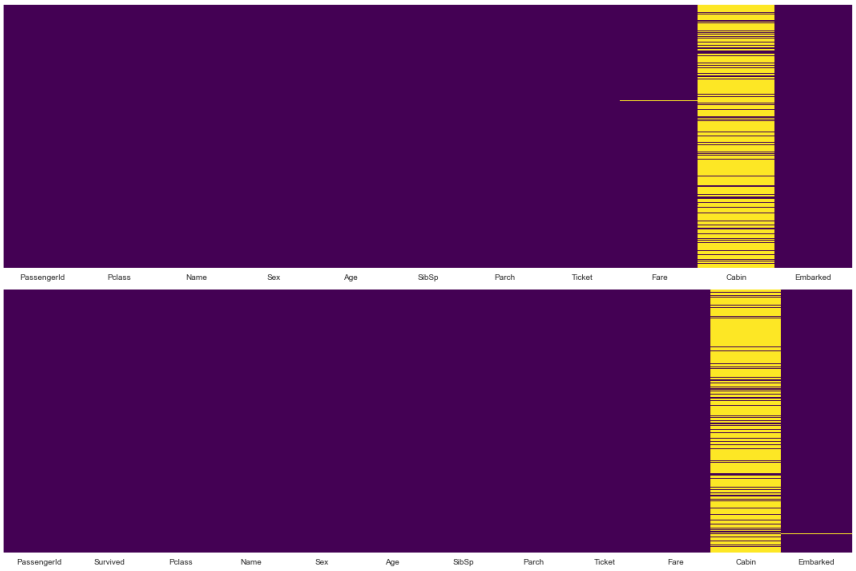
I tried to fill the missing values based of my age columns for both the data set (Trainind & Test Set)

In [30]:
```
dftrain.head(1)
```

Out[30]:

	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	7.25	NaN	S

In [31]:
```
fig, (ax1,ax2) = plt.subplots (2, figsize = (15,10))
sns.heatmap(dftest.isnull(), cmap="viridis", yticklabels=False
            ,cbar = False, ax= ax1)
sns.heatmap(dftrain.isnull(), cmap="viridis", yticklabels=False
            ,cbar = False, ax= ax2)
plt.tight_layout()
```



In [32]:
```
dftest.head(1)
```

Out[32]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q

5.B. ENCODING: Categorical Data

In [33]:
```
sex = pd.get_dummies(dftrain["Sex"], drop_first=True)
embark = pd.get_dummies(dftrain["Embarked"], drop_first=True)
clss = pd.get_dummies(dftrain["Pclass"], drop_first=True)


sext = pd.get_dummies(dftest["Sex"], drop_first=True)
embarkt = pd.get_dummies(dftest["Embarked"], drop_first=True)
clsst = pd.get_dummies(dftest["Pclass"], drop_first= True)
```

In [34]:
```
# Concatinating the new features to the Clear Trainig set:
train = pd.concat([dftrain, sex, embark, clss], axis =1)
train.head(1)
```

Out[34]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	male	Q	S	2	3
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25	NaN	S	1	0	1	0	1

In [35]:
```
# Concatinating the new features to the Test set:
test = pd.concat([dftest, sext, embarkt, clsst], axis =1)
test.head(1)
```

Out[35]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	male	Q	S	2	3
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q	1	1	0	0	1

In [36]:
```
# Dropping columns which are not required from training set
train.drop(["PassengerId", "Pclass","Name", "Sex", "Ticket", "Cabin", "Embarked"], axis = 1,inplace = True)
train.head(1)
```

Out[36]:

	Survived	Age	SibSp	Parch	Fare	male	Q	S	2	3
0	0	22.0	1	0	7.25	1	0	1	0	1

In [37]:
```
# Dropping columns which are not required from training set
test.drop(["PassengerId", "Pclass","Name", "Sex", "Ticket", "Cabin", "Embarked"], axis = 1,inplace = True)
test.head(1)
```

Out[37]:

	Age	SibSp	Parch	Fare	male	Q	S	2	3
0	34.5	0	0	7.8292	1	1	0	0	1

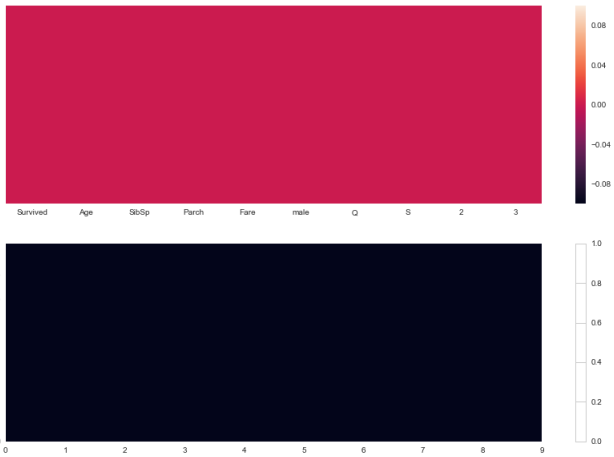
In [ ]:
```

```

In [38]:
```
fig, (ax1, ax2) = plt.subplots(2, figsize = (15,10))
sns.heatmap(train.corr(), annot= True, cmap= "magma", ax=ax1)
sns.heatmap(test.corr(), annot= True, cmap= "magma", ax=ax2)

plt.show()
plt.tight_layout()
```


```
<matplotlib.figure.Figure at 0x1a230d4240>
```

In [39]:
```
fig, (ax1, ax2) = plt.subplots(2, figsize = (15,10))
sns.heatmap(train.isnull(), ax=ax1, yticklabels=False)
sns.heatmap(test.isnull(), yticklabels=False)
plt.show()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-39-f6a9988a00f1> in <module>()
      1 fig, (ax1, ax2) = plt.subplots(2, figsize = (15,10))
      2 sns.heatmap(train.isnull(), ax=ax1, yticklabels=False)
----> 3 sns.heatmap(test.isnull(), yticklabels=False)
      4 plt.show()

~/anaconda3/lib/python3.6/site-packages/seaborn/matrix.py in heatmap(data, vmin, vmax, cmap, center, robust, annot, fmt, annot_kws, linewidths, linecolor, cbar, cbar_kws, cbar_ax, square, xticklabels, yticklabels, mask, ax, **kwargs)
    526     if square:
    527         ax.set_aspect("equal")
--> 528     plotter.plot(ax, cbar_ax, kwargs)
    529     return ax
    530

~/anaconda3/lib/python3.6/site-packages/seaborn/matrix.py in plot(self, ax, cax, kws)
    290         # Possibly add a colorbar
    291         if self.cbar:
--> 292             cb = ax.figure.colorbar(mesh, cax, ax, **self.cbar_kws)
    293             cb.outline.set_linewidth(0)
    294             # If rasterized is passed to pcolormesh, also rasterize the

~/anaconda3/lib/python3.6/site-packages/matplotlib/figure.py in colorbar(self, mappable, cax, ax, use_gridspec, **kw)
   1862                 cax, kw = cbar.make_axes(ax, **kw)
   1863         cax._hold = True
-> 1864         cb = cbar.colorbar_factory(cax, mappable, **kw)
   1865
   1866         self.sca(current_ax)

~/anaconda3/lib/python3.6/site-packages/matplotlib/colorbar.py in colorbar_factory(cax, mappable, **kwargs)
   1366         cb = ColorbarPatch(cax, mappable, **kwargs)
   1367     else:
-> 1368         cb = Colorbar(cax, mappable, **kwargs)
   1369
   1370     cid = mappable.callbacksSM.connect('changed', cb.on_mappable_changed)

~/anaconda3/lib/python3.6/site-packages/matplotlib/colorbar.py in __init__(self, ax, mappable, **kw)
    944                 kw['alpha'] = mappable.get_alpha()
    945
--> 946             ColorbarBase.__init__(self, ax, **kw)
    947
    948     def on_mappable_changed(self, mappable):

~/anaconda3/lib/python3.6/site-packages/matplotlib/colorbar.py in __init__(self, ax, cmap, norm, alpha, values, boundaries, orientation, ticklocation, extend, spacing, ticks, format, drawedges, filled, extendfrac, extendrect, label)
    327         # The rest is in a method so we can recalculate when clim changes.
    328         self.config_axis()
--> 329         self.draw_all()
    330
    331     def _extend_lower(self):

~/anaconda3/lib/python3.6/site-packages/matplotlib/colorbar.py in draw_all(self)
    349         '''
    350
--> 351         self._process_values()
    352         self._find_range()
    353         X, Y = self._mesh()

~/anaconda3/lib/python3.6/site-packages/matplotlib/colorbar.py in _process_values(self, b)
    702                 self.norm.vmin,
    703                 self.norm.vmax,
--> 704                 expander=0.1)
    705
    706             b = self.norm.inverse(self._uniform_y(self.cmap.N + 1))

~/anaconda3/lib/python3.6/site-packages/matplotlib/transforms.py in nonsingular(vmin, vmax, expander, tiny, increasing)
   2911         vmax = expander
   2912
-> 2913     elif vmax - vmin <= maxabsvalue * tiny:
   2914         if vmax == 0 and vmin == 0:
   2915             vmin = -expander

TypeError: numpy boolean subtract, the `-` operator, is deprecated, use the bitwise_xor, the `^` operator, or the logical_xor function instead.
```



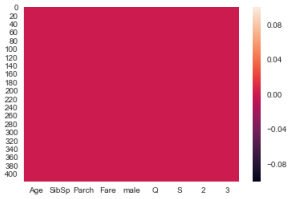
5.C. Creating Train & Test Split

In [ ]:
```
```

In [41]:
```
test_y = pd.read_csv("gender_submission.csv")
X_train = train.drop("Survived", axis=1)
y_train = train["Survived"]
X_test = test.fillna(dftest["Fare"].mean())
y_test = test_y["Survived"]
```

In [42]:
```
sns.heatmap(X_test.isnull())
```

Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0x1a185f3f98>



6. Modeling

6. a. Logistic Regression

In [43]:
```
log = LogisticRegression()
log.fit(X_train, y_train)
pred = log.predict(X_test)
```

6.b. Random Forest

In [44]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
pred1 = rf.predict(X_test)
```

In [ ]:
```

```

7. Model Evaluation

In [45]:
```
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
```

In [46]:
```
# Report of Logistic Regression
print(confusion_matrix(y_test, pred))
print(classification_report(y_test, pred))
```

```
[[257   9]
 [ 11 141]]
              precision    recall  f1-score   support

           0       0.96      0.97      0.96       266
           1       0.94      0.93      0.93       152

avg / total       0.95      0.95      0.95       418
```

In [ ]:
```

```

In [47]:
```
# Report of Random Forest
print(confusion_matrix(y_test, pred1))
print(classification_report(y_test, pred1))
```

```
[[230  36]
 [ 45 107]]
              precision    recall  f1-score   support

           0       0.84      0.86      0.85       266
           1       0.75      0.70      0.73       152

avg / total       0.80      0.81      0.80       418
```

Next step in order to increase the precision and get more accuracy. I will be doing more feature engineering such as trying to grab the title of the names, cabin letter and ticket information.

In [ ]:
```

```

In [48]:
```
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
subm = rf.predict(X_test)
```

In [49]:
```
sub_rep = pd.DataFrame({"PassengerId": dftest["PassengerId"],
                        "Survived" : subm})
```

In [50]:
```
sub_rep.to_csv("Tit Sub.csv", index = False)
print (sub_rep.shape)
```

```
(418, 2)
```

In [ ]:
```

```

In [ ]:
```

```