Famous Salami, salamifamous@gmail.com

Titanic Dataset: Exploratory Data Analysis and Visualization

In [ ]:

In [ ]:
```
import pandas as pd
import  matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import re
colors = ['#36454F', '#DC143C', '#fdaa48','#6890F0','#A890F0']
```

Exploratory Data Analysis

In [ ]:
```
data = pd.read_csv("Titanic Dataset.csv")
raw_data = pd.read_csv("Titanic Dataset.csv")
data.head(6)
```

Out[ ]:

	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
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q

In [ ]:
```
data.info()
```

```
<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
```

In [ ]:
```
data.isna().sum()
```

Out[ ]:
```
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 [ ]:
```
survivor_group = data['Survived'].value_counts()
print(survivor_group)
print()
prcnt_surv = (survivor_group[1] / data.shape[0]) * 100
prcnt_n_surv = (survivor_group[0] / data.shape[0]) * 100
```

```
print(f'Percentage of survivors is {round(prcnt_surv)}%')
print(f'Percentage of non-survivors is {round(prcnt_n_surv)}%')
```

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

Percentage of survivors is 38%
Percentage of non-survivors is 62%
```

Observations

- The dataset shows information about the passengers on board the Titanic ship. It comprises of 891 observations (rows) and features (columns), representing passengers.
- There are missing values in three major entries, which are 'Age', 'Cabin' and 'Port of Embarkation'; the missing values are listed below:

  'Cabin' has 687 missing values
  'Age' has 177 missing values
  'Embarkation' has 2 missing values

- The dataset records a total of 342 survivors, representing 38% of the entire passengers, while 549, 62% died in the shipwreck.

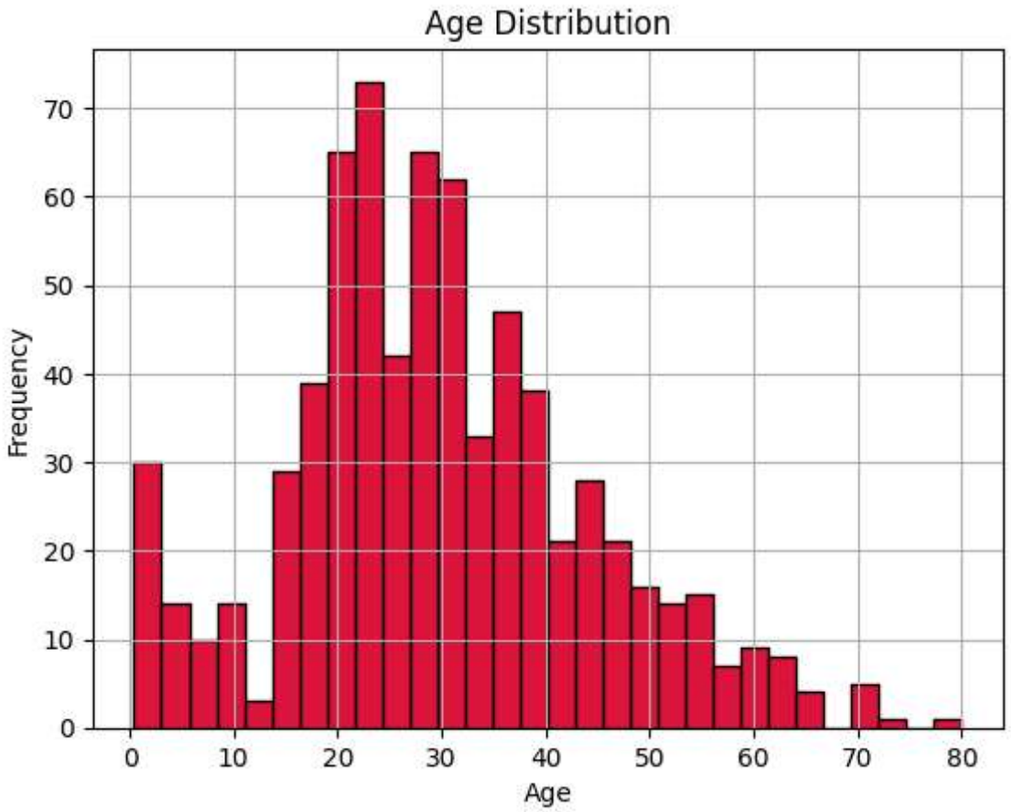
Data Cleaning and Preprocessing

In [ ]:
```
# Embarkation Port has 2 missing data, this would be filled using the forward fill method
data['Embarked'] = data['Embarked'].ffill()
missing_embarkation_rows = data['Embarked'].isna().sum()
print(f'Missing embarkation port after cleaning is: {missing_embarkation_rows}')
```

```
Missing embarkation port after cleaning is: 0
```

In [ ]:
```
data['Age'].hist(bins=30, color=colors[1], edgecolor='black')

plt.title('Age Distribution')
plt.xlabel('Age')
```

```
plt.ylabel('Frequency')
plt.show()
```



- The histogram plot of the Age distribution is right-skewed, this implies the median is the most accurate value to use in filling the missing age
- I would use a more suitable approach to address missing age values by first extracting passenger titles. Then, I'll fill the missing age values with the median age corresponding to each title group.

In [ ]:
```
title_conditions = [
    data['Name'].str.contains(r'\bMr\b', case=False, regex=True),
    data['Name'].str.contains(r'\bMrs\b', case=False, regex=True),
    data['Name'].str.contains(r'\bMiss\b', case=False, regex=True)
]
```

```
titles = ['Mr', 'Mrs', 'Miss']
data['Title'] = np.select(title_conditions, titles, default='NA') # Engineering a new feature, `Title`
```

In [ ]:
```
# Age has 177 missing data, these would be filled with the median age of each title group
median_age_by_title = data.groupby('Title')['Age'].median()

data['Age'] = data.apply(lambda row: median_age_by_title[row['Title']] if pd.isnull(row['Age']) else row['Age'], axis=1)

missing_age_data = data['Age'].isna().sum()
print(f'Missing age entries after cleaning is: {missing_age_data}')
```

```
Missing age entries after cleaning is: 0
```

In [ ]:
```
# The Cabin feature has too many missing data, representing 77% of the dataset, it is therefore better dropped [in this context]
data = data.drop(columns='Cabin')
data.info()
```

```
<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          891 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  Embarked     891 non-null    object 
 11  Title        891 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

Preprocessing

In [ ]:
```
# Preprocessing
data['Age'] = data['Age'].astype(int)
```

```
data['Fare'] = data['Fare'].round(2)
data['Pclass'] = data['Pclass'].astype(int)
```

In [ ]:
```
# Grouping by age to create the 'Age Group' feature/column
def age_grouping(age):
    if age < 18:
        return 'Child'
    elif age < 25:
          return 'Young Adult'
    elif age < 65:
          return 'Adult'
    else:
        return 'Senior'

data['AgeGroup'] = data['Age'].apply(age_grouping)
data['AgeGroup'].value_counts()
```

Out[ ]:
```
AgeGroup
Adult          561
Young Adult    201
Child          118
Senior          11
Name: count, dtype: int64
```

In [ ]:
```
# Grouping by Sex to create the 'MWBG' (ManWomanGirlBoy) feature
def gender_grouping(title, age_group, sex):
    if title == 'Mr':
        return 'Man'
    elif title == 'Mrs':
        return 'Woman'
    elif title == 'Miss':
        return 'Woman' if age_group != 'Child' else 'Girl'
    else:
        if age_group == 'Child':
            return 'Boy' if sex == 'male' else 'Girl'
        else:
            return 'Man' if sex == 'male' else 'Woman'

data['MWBG'] = data.apply(lambda row: gender_grouping(row['Title'], row['AgeGroup'], row['Sex']), axis=1)
data['MWBG'].value_counts()
```

Out[ ]:
```
MWBG
Man      536
Woman    263
Girl      51
Boy       41
Name: count, dtype: int64
```

In [ ]:
```
# Grouping by travel partner ('SibSp' or 'Parch') to create the 'TravelPartner' feature
def travel_partner(sibsp, parch):
    co_traveler = sibsp + parch
    if co_traveler > 0:
        return 'With family'
    else:
        return 'Alone'

data['TravelPartner'] = data.apply(lambda row: travel_partner(row['SibSp'], row['Parch']), axis=1)
```

In [ ]:
```
data.head()
```

Out[ ]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Embarked	Title	AgeGroup	MWBG	TravelPartner
0	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.25	S	Mr	Young Adult	Man	With family
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38	1	0	PC 17599	71.28	C	Mrs	Adult	Woman	With family
2	3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.92	S	Miss	Adult	Woman	Alone
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.10	S	Mrs	Adult	Woman	With family
4	5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.05	S	Mr	Adult	Man	Alone

Notes

The missing values were replaced for the 'Age' and 'Embarked' features while the 'Cabin' feature was dropped as it has too many missing values, this decision, however, is contextual.
This feature would still be useful in the analysis section, hence it's stored in another dataframe.

- The missing 'Age' values were filled using the median value of each title group and then preprocessed from float values to integer values
- The missing 'Embarked' values were filled using the forward fill method
- The missing 'Cabin' feature was dropped as being too much (77%) and keeping this can give distorted analysis.

Some feature engineering tasks were also carried out, where I extracted four new features `Title`, `AgeGroup`, `MWBG` and `TravelPartner` from the 'Age', 'Sex' an 'Name' columns. These gave more insights about the data and would be useful in the Data Analysis section of this presentation.

Data Analysis

In [ ]:
```
def addCaption(dataframe, typ='df', axes = None, ax = None, xtick = None):
    if typ == 'df':
        total = dataframe.sum(axis=1)  # Total count in each group
        if axes is not None:
            for i, ax in enumerate(axes):
                percent_label = dataframe.iloc[:, i] / dataframe.iloc[:, i].sum() * 100
                for bar in ax.containers:
                    ax.bar_label(bar, labels=np.round(percent_label).apply(lambda x: f'{x}%'), label_type='edge', bbox=dict(facecolor='white', edgecolor='bla
                    if (xtick): ax.set_xticklabels(xtick)
        else:
            for i, bar in enumerate(ax.containers):
                percent_label = dataframe.iloc[:, i] / total * 100  # Calculate percentage based on total
                ax.bar_label(bar, labels=np.round(percent_label).apply(lambda x: f'{x}%'), label_type='edge', bbox=dict(facecolor='white', edgecolor='black',
        return
    else:
        percent_label = dataframe / dataframe.sum() * 100
        for i, bar in enumerate(ax.containers):
            ax.bar_label(bar, labels=np.round(percent_label).apply(lambda x: f'{x}%'), label_type='edge', bbox=dict(facecolor='white', edgecolor='black', box
        if xtick:
            ax.set_xticklabels(xtick)
        return
```

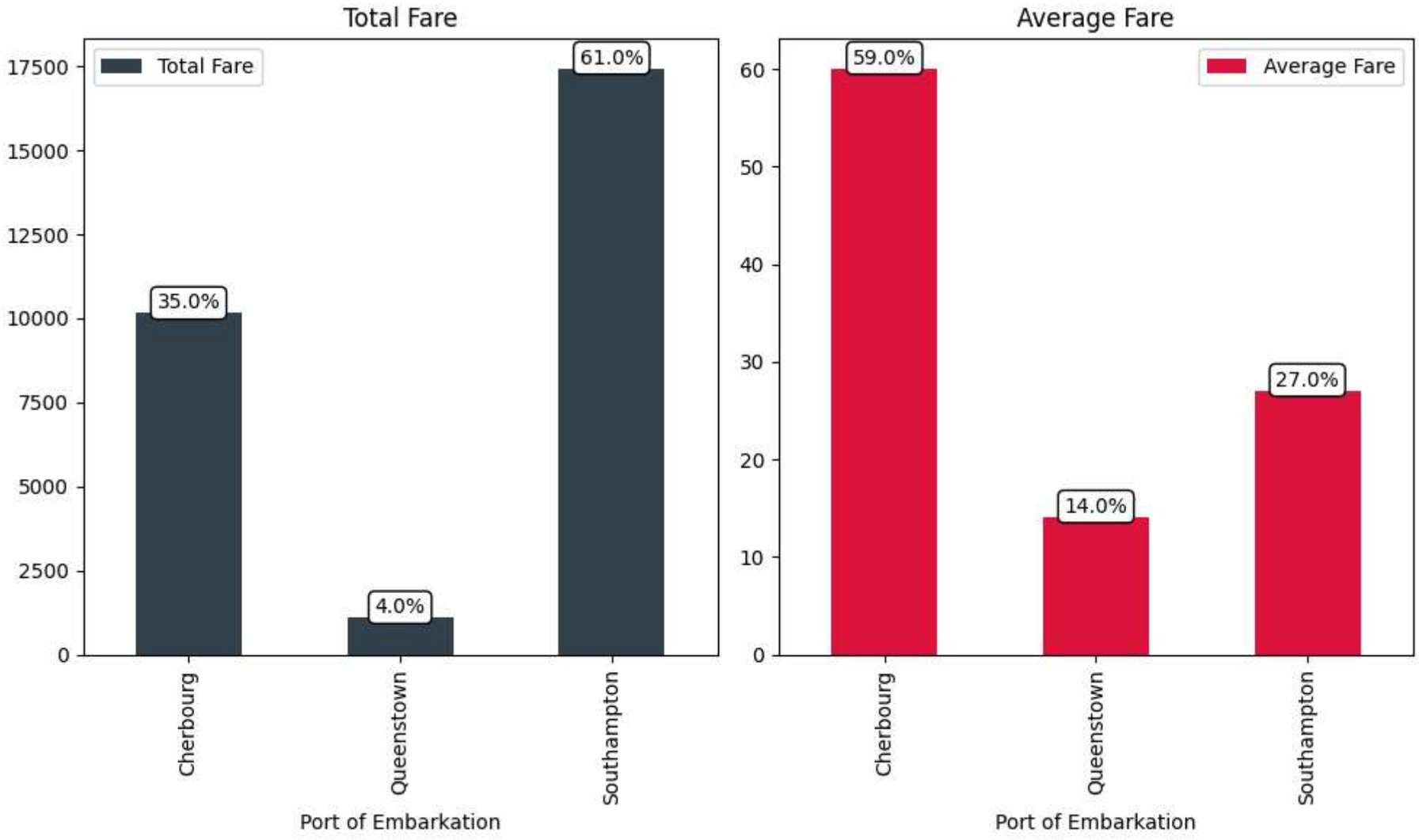
#1a) Analysis of Trip Fare by Port of Embarkation

In [ ]:
```
embarkation_fare = data.groupby('Embarked')['Fare'].sum()
embarkation_passenger = data.groupby('Embarked')['Fare'].size()
avg_embarkation_fare = data.groupby('Embarked')['Fare'].mean()

#embarkation_dataframe = pd.DataFrame({'Total Fare': embarkation_fare, 'Number of Passengers': embarkation_passenger, \
 #                   'Average Fare': round(avg_embarkation_fare, 2)})
embarkation_dataframe = pd.DataFrame({'Total Fare': embarkation_fare, 'Average Fare': round(avg_embarkation_fare, 2)})
embarkation_dataframe
```

Out[ ]:

Embarked	Total Fare	Average Fare
C	10152.24	60.07
Q	1102.23	14.13
S	17439.62	27.08

In [ ]:
```
fig, axes = plt.subplots(1,2, figsize=(10,6))
embarkation_dataframe.plot(kind='bar', subplots=True, color=colors, xlabel="Port of Embarkation", ax=axes)
addCaption(embarkation_dataframe, 'df', axes, None, ['Cherbourg', 'Queenstown', 'Southampton'])
plt.tight_layout()
plt.show()
```



- The Southampton (S) Port recorded the highest sum of ticket sales at $17,439.62, representing 61\% of the total, followed by the Cherbourg (C) Port at$10,152.24 (35%) with Queenstown (Q) Port recording the least sales overall of $1,102.23 (4%).

- The ticket seems to be more expensive at the Cherbourg Port at the average cost of $60, followed by the Southampton Port with average ticket cost of $27, while the average fare at Queenstown was $14

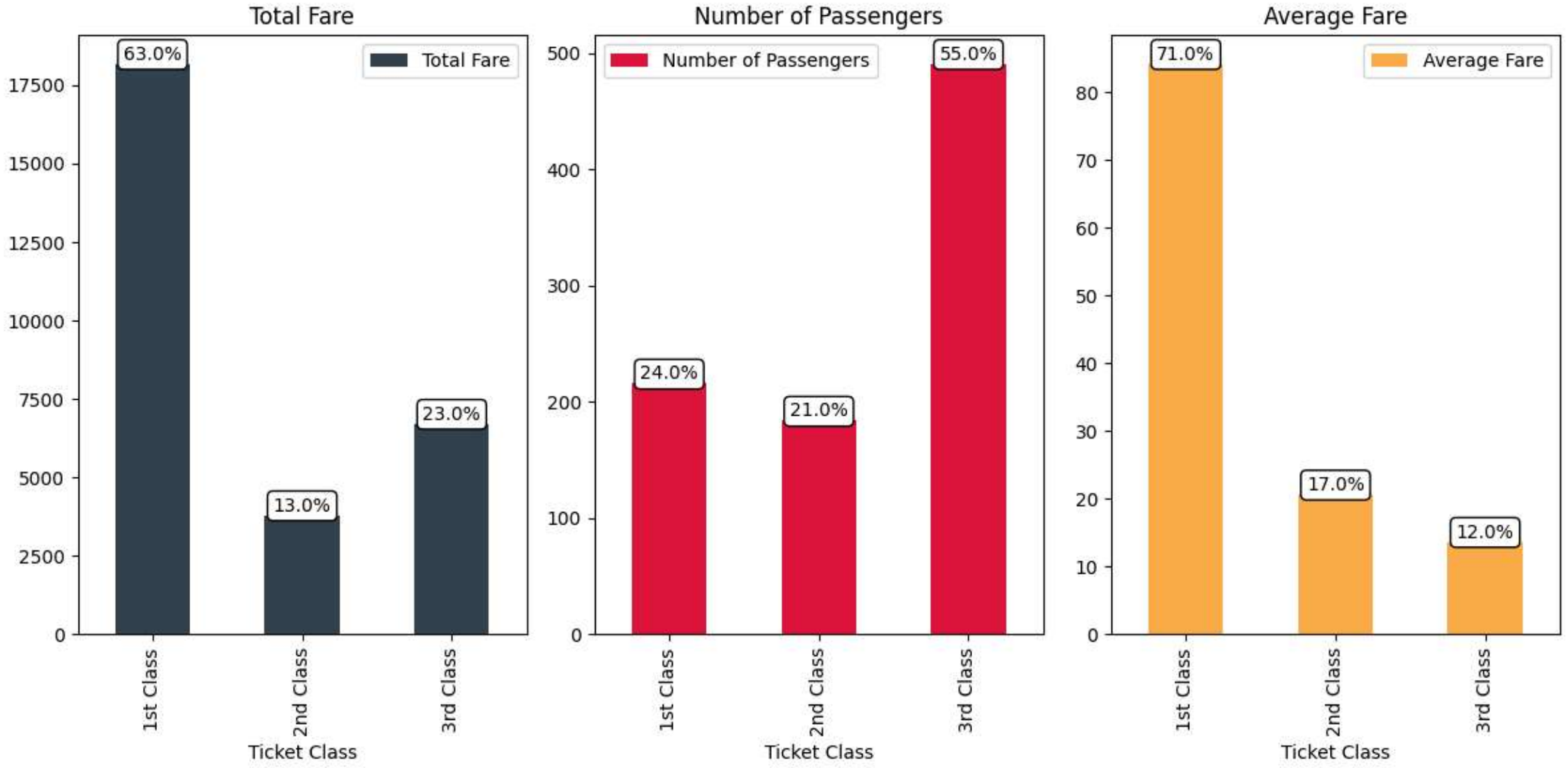
#1b) Analysis of Trip Fare by Ticket Class

In [ ]:
```
fare_by_pclass = data.groupby('Pclass')['Fare'].sum()
passenger_by_pclass = data.groupby('Pclass')['Fare'].size()
avg_fare_by_pclass = data.groupby('Pclass')['Fare'].mean()

data_by_pclass = pd.DataFrame({'Total Fare': fare_by_pclass, 'Number of Passengers': passenger_by_pclass, \
                  'Average Fare': round(avg_fare_by_pclass, 2)})
data_by_pclass
```

Out[ ]:

Pclass	Total Fare	Number of Passengers	Average Fare
1	18177.40	216	84.15
2	3801.84	184	20.66
3	6714.85	491	13.68

In [ ]:
```
fig, axes = plt.subplots(1,3, figsize=(12,6))
data_by_pclass.plot(kind='bar', subplots=True, color=colors, xlabel="Ticket Class", ax=axes)
addCaption(data_by_pclass, 'df', axes, None, ['1st Class', '2nd Class', '3rd Class'])
plt.tight_layout()
plt.show()
```



- The highest sales ($18,177.4at63$6,714.85 at 23%
  while the least sales was with the Second Class ticket at the sum of $3,801 at 13%.
- An average First Class ticket cost as high as $84,followedbySecondClassticketat$20.66 and Third Class ticket at $13.68

- A total of 216 passengers boarded the First Class ticket, while 184 passengers were recorded for Second Class with Third Class having the highest number of passengers of 491.

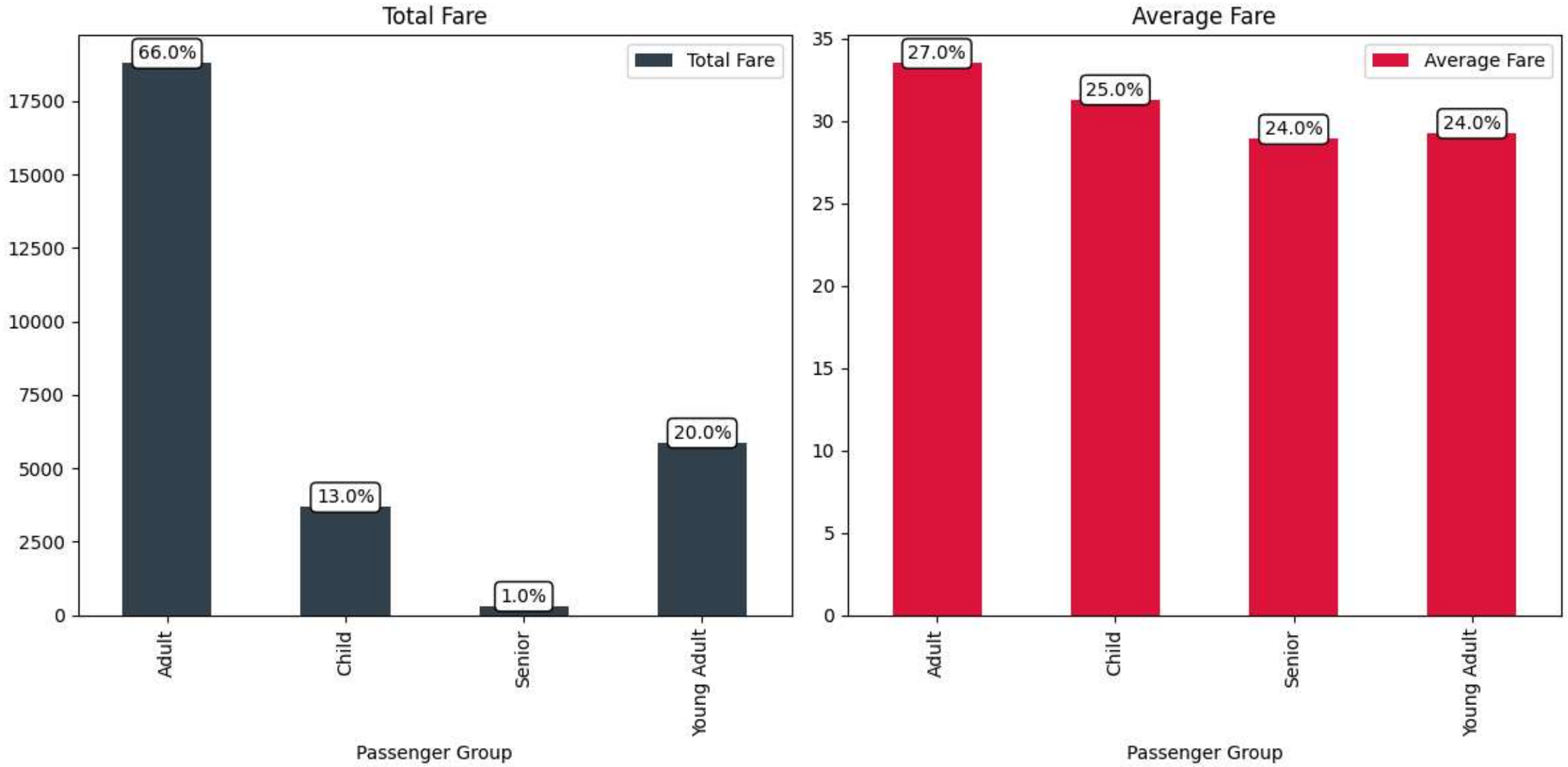
#1c) Analysis of Trip Fare by Age

In [ ]:
```
fare_by_sex = data.groupby('AgeGroup')['Fare'].sum()
passenger_by_sex = data.groupby('AgeGroup')['Fare'].size()
avg_fare_by_sex = data.groupby('AgeGroup')['Fare'].mean()

data_by_sex = pd.DataFrame({'Total Fare': round (fare_by_sex,2), 'Average Fare': round(avg_fare_by_sex, 2)})
data_by_sex
```

Out[ ]:

AgeGroup	Total Fare	Average Fare
Adult	18801.75	33.51
Child	3693.10	31.30
Senior	317.96	28.91
Young Adult	5881.28	29.26

In [ ]:
```
fig, axes = plt.subplots(1,2, figsize=(12,6))
data_by_sex.plot(kind='bar', subplots=True, color=colors, xlabel="Passenger Group", ax=axes)
addCaption(data_by_sex, 'df', axes)
plt.tight_layout()
plt.show()
```

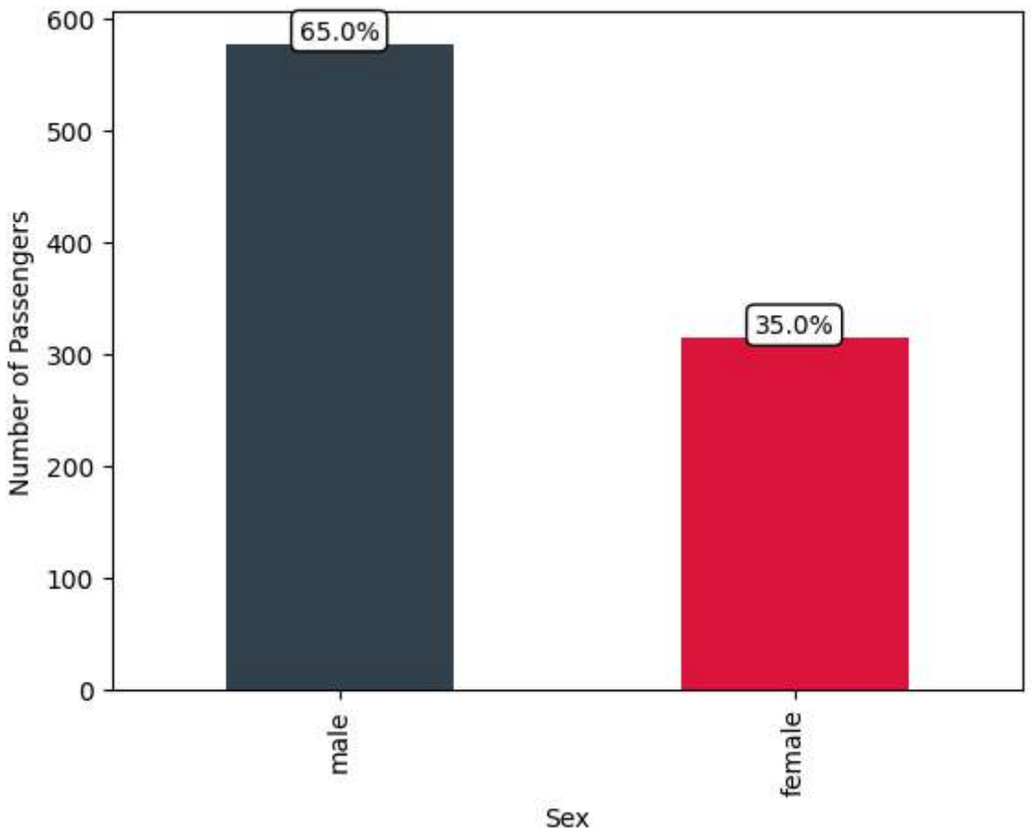


- The 'AgeGroup' feature was engineered from the 'Age' feature, with passengers with age less than 16 counted as children, 16 - 25 as Young Adult, 26 - 65 as Adult and 65 and above as Senior citizens.
- The age group comprises of `Senior Citizens`, `Adult`, `Young Adult` and `Children` set.

- The bar plot shows the highest ticket purchase was made by the Adult age group at a total of $18,801.75, this is followed by the Young Adult group with a total of $5,881.28 ticket purchase, the children are the third in this category with ticket sales of $3,693.10, while a very few Senior citizens partook in the trip.

#2a) Analysis of Passengers by Age and Gender

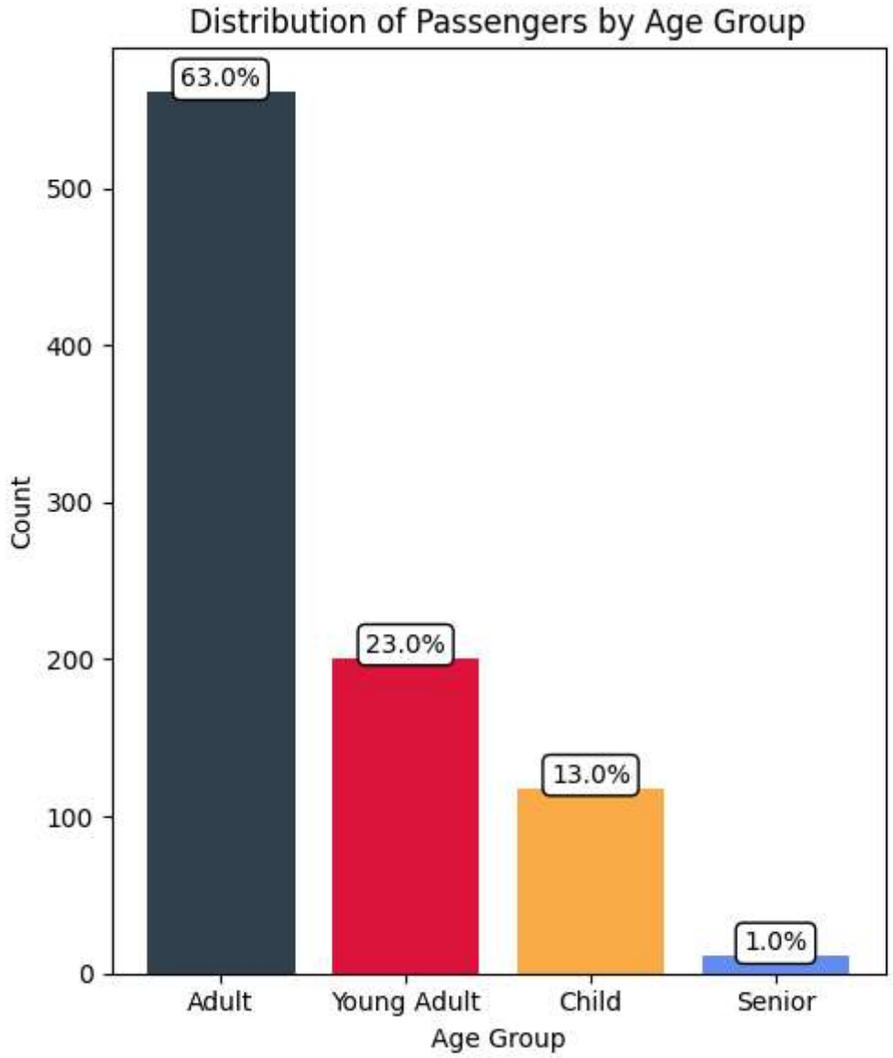
In [ ]:
```
ax = data['Sex'].value_counts().plot(kind="bar", ylabel='Number of Passengers', color=colors)
addCaption(data['Sex'].value_counts(), 'sr',  None, ax)
```

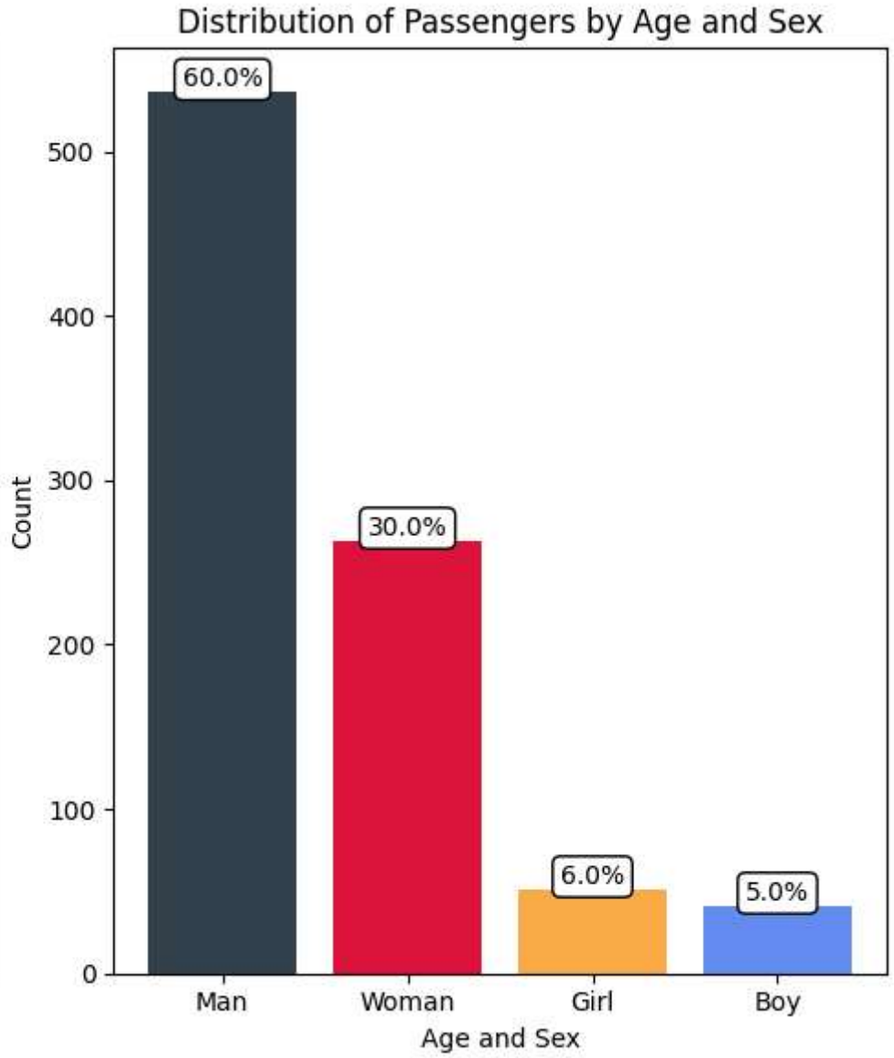


- The Titanic trip was dominated by male passengers, amounting to 65% of the total passengers, representing almost double the female numbers, while female passengers represent 35% of the passengers.

In [ ]:

```
age_group = data['AgeGroup'].value_counts()
gender_group = data['MWBG'].value_counts()

fig, ax = plt.subplots(1, 2, figsize=(10, 6))

ax[0].bar(age_group.index, age_group.values, color=colors)
ax[0].set_title('Distribution of Passengers by Age Group')
ax[0].set_xlabel('Age Group')
ax[0].set_ylabel('Count')
addCaption(age_group, 'sr',  None, ax[0])


ax[1].bar(gender_group.index, gender_group.values, color=colors)
ax[1].set_title('Distribution of Passengers by Age and Sex')
ax[1].set_xlabel('Age and Sex')
ax[1].set_ylabel('Count')
addCaption(gender_group, 'sr',  None, ax[1])

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

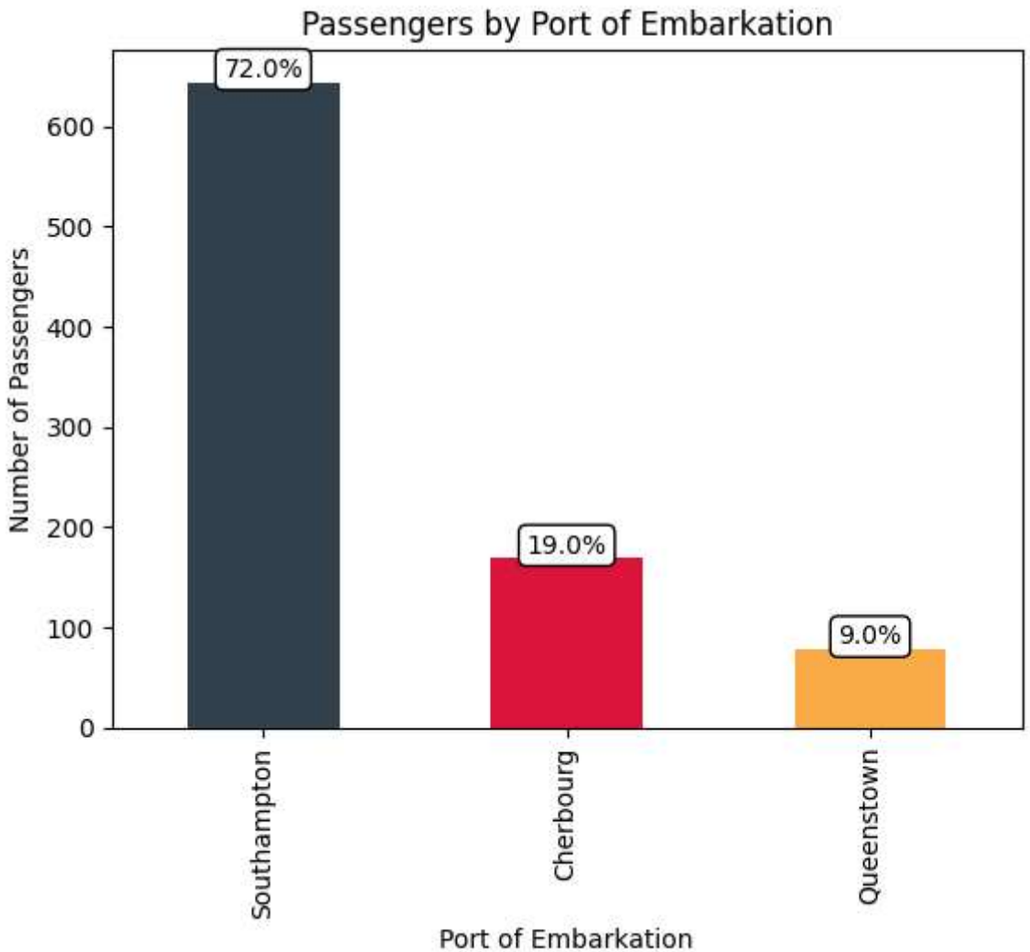




- The distribution by age group reveals that the Adult age group made largest number of the passengers at 63% of the overall passengers, followed by the Young Adults representing 23% of the passengers, children made 13% and Senior citizens were so few at 1%.
- The distribution by sex shows that men formed 60% of the total passengers, followed by women at 30%, boys made 5% of the passengers, while girls formed 6%.

#2b) Analysis of Passengers by Port of Embarkation

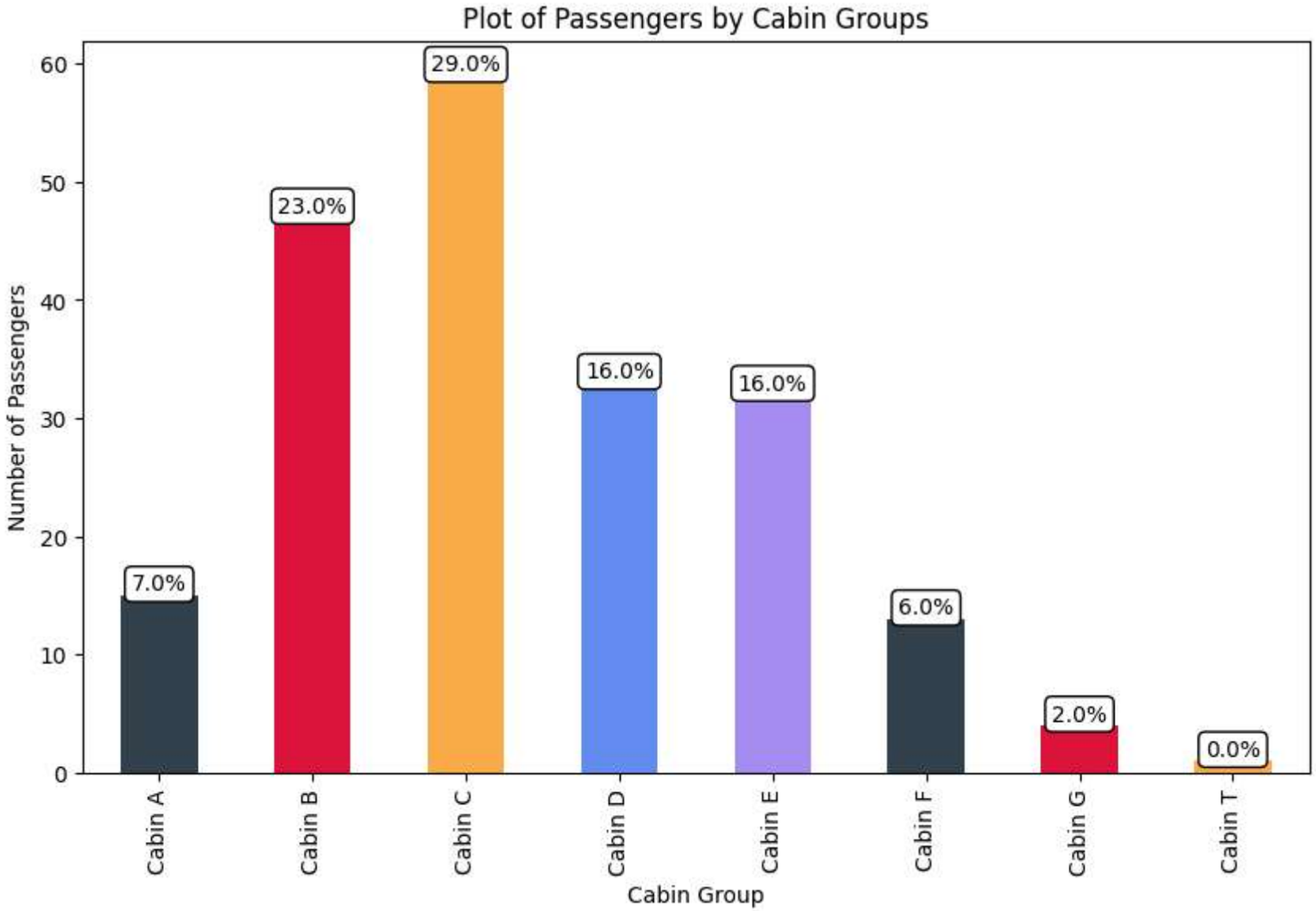
In [ ]:
```
survival_by_port = data['Embarked'].value_counts()
#survival_group = survival_by_port.rename(columns={0: 'Not Survived', 1: 'Survived'})
ax = survival_by_port.plot(kind="bar", title="Passengers by Port of Embarkation", xlabel='Port of Embarkation', \
                      ylabel='Number of Passengers', color=colors)
ax.set_xticklabels(['Southampton', 'Cherbourg', 'Queenstown'])
addCaption(survival_by_port, 'sr', None, ax, None)
```



- The Southampton Port recorded the largest number of passengers, 72% of the passengers boarded from this port, followed by the Cherbourg Port with 19% of the passengers with Queenstown recording the list number of passengers at 9%

#2c) Analysis of Passengers by Cabin Group

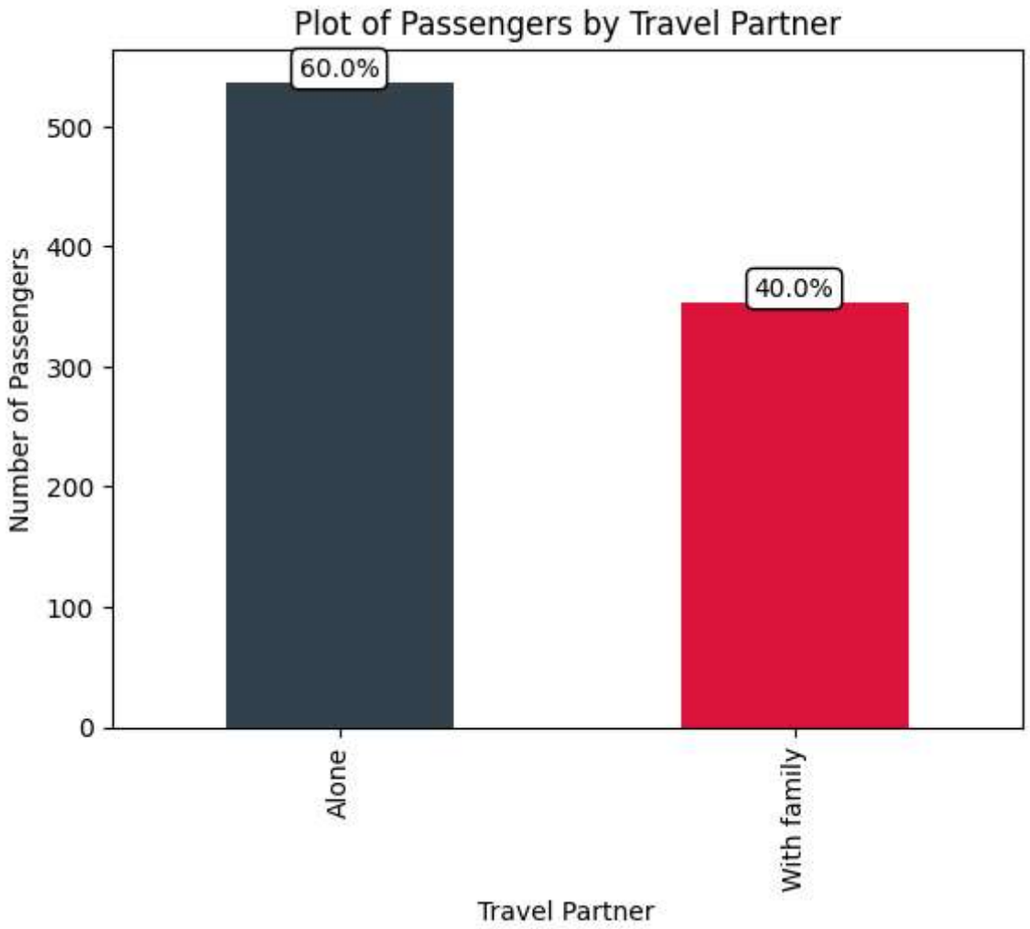
In [ ]:
```
# This is not good for analysis
# I added it just in case there was any dataset that has the Cabin column intact or...
# for any other purpose not mentioned.

cabin_counts = raw_data['Cabin'].value_counts()
first_letter = cabin_counts.index.str[0]

cabin_grp_counts = cabin_counts.groupby(first_letter).sum()

ax = cabin_grp_counts.plot(kind="bar", title="Plot of Passengers by Cabin Groups", figsize= (10, 6), color=colors, \
                      ylabel='Number of Passengers', xlabel='Cabin Group')
ax.set_xticklabels(['Cabin {}'.format(label.get_text()) for label in ax.get_xticklabels()])
addCaption(cabin_grp_counts, 'sr', None, ax)
```



- This analysis is based on the 204 Cabin numbers captured out of the 891 passengers in the dataset.
- Majority of the passengers occupied Cabin C in the ship, followed by Cabin B, Cabin D and E with equal number of passengers. The number of passengers shows 60 as its peak, this is due to over 70% of the Cabin information missing from the dataset.

#2d) Analysis of Passengers by Travel Partner

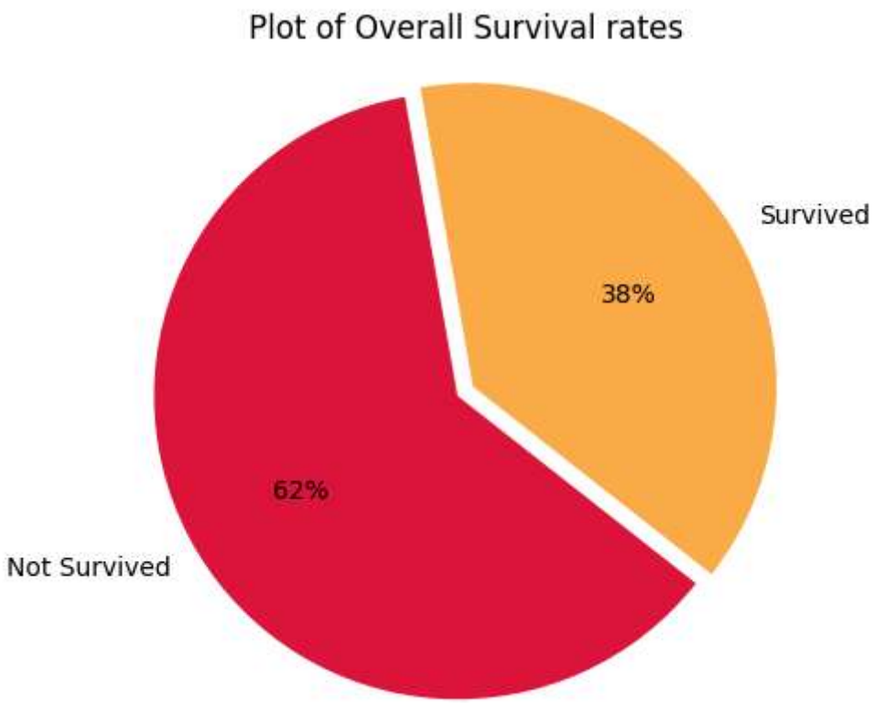
In [ ]:
```
ax = data['TravelPartner'].value_counts().plot(kind="bar", title='Plot of Passengers by Travel Partner', \
                                               xlabel='Travel Partner', ylabel='Number of Passengers', color=colors)
addCaption(data['TravelPartner'].value_counts(), 'sr', None, ax)
```



- The passengers were grouped into two classes, those who traveled alone, represnting 60%, while passengers who traveled with either parent and child or sibling and spouse represent the remaining 40%

#3a) Analysis of Overall Survival rate

In [ ]:
```
survivor_group = data.groupby('Survived').size()
explode =(0, 0.06)
labels = ['Not Survived', 'Survived']
svplot = survivor_group.plot(kind="pie", explode=explode, autopct='%1.0f%%', labels=labels, startangle=100, \
                             title="Plot of Overall Survival rates", colors=[colors[1],colors[2]])
#svplot.set_xticklabels(['Not survived', 'Survived'])
plt.axis('equal')
addCaption(survivor_group, 'sr', None, svplot)
plt.show()
```



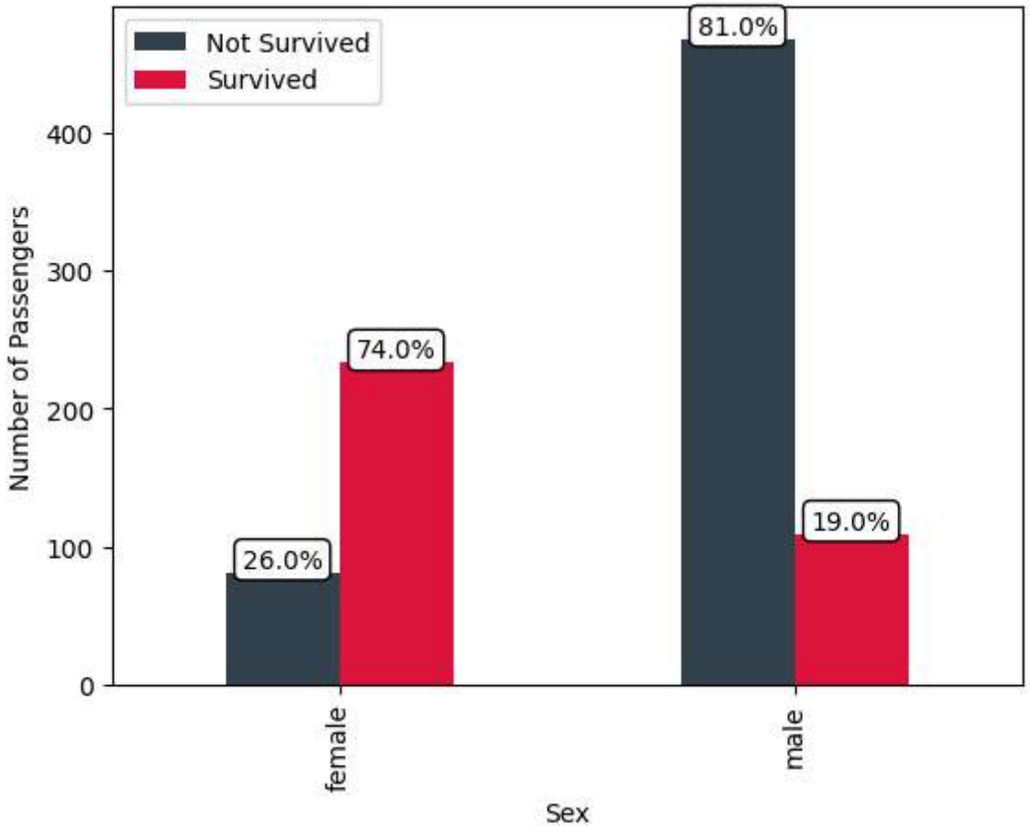
- A very few number of passengers appear to have survived the Titanic shipwreck. 38% passengers survived, while 62% of the passengers never survived the wreck.

#3b) Analysis of Survival rate by Genger

In [ ]:
```
survival_by_gender = data.groupby(['Sex','Survived']).size().unstack()
ax = survival_by_gender.plot(kind="bar", ylabel='Number of Passengers', color=colors)
ax.legend(['Not Survived', 'Survived'])
survival_by_gender
addCaption(survival_by_gender,'df', None, ax)
survival_by_gender
```

Out[ ]:

Survived	0	1
Sex		
female	81	233
male	468	109

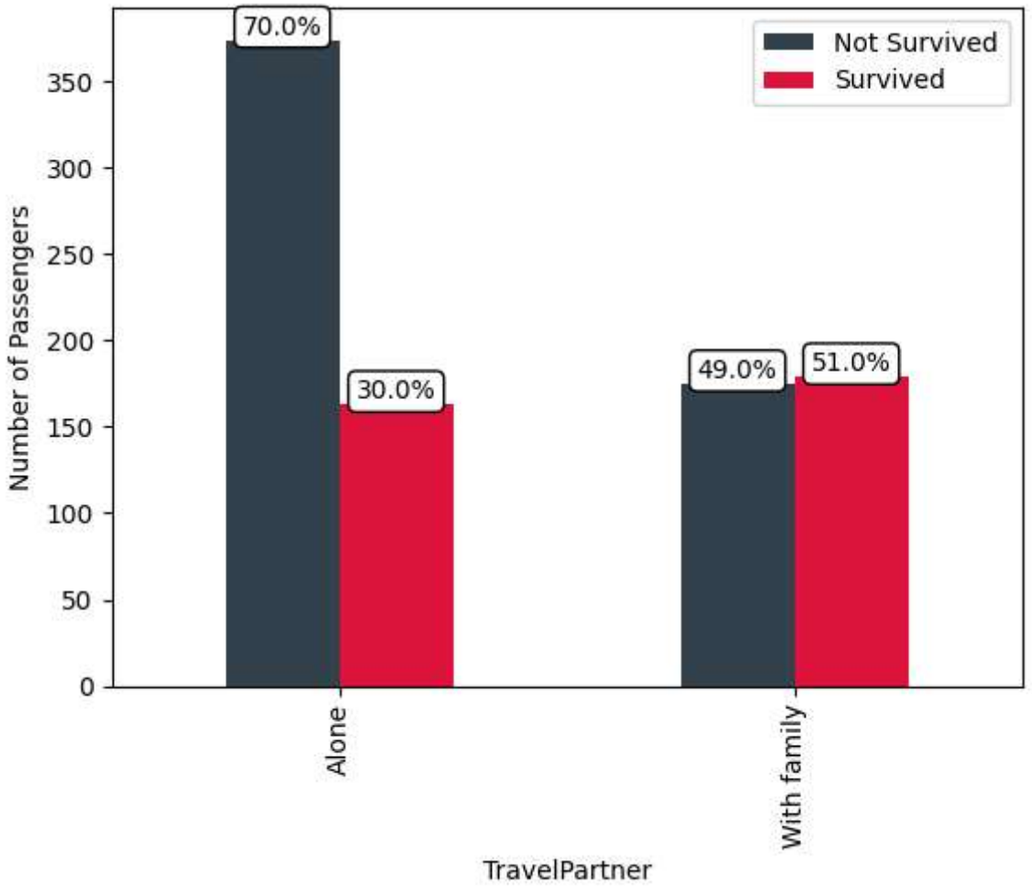


- The female gender had higher chances of survival, 74% of the female passengers, totalling 233 survived the Titanic wreck, while 19% of the male passengers, totalling 109 survived.
- 81% of the males and 26% of the females did not survive the Titanic.

#3c) Analysis of Survival rate by Travel Partner

In [ ]:
```
survival_by_partner = data.groupby(['TravelPartner','Survived']).size().unstack()
ax = survival_by_partner.plot(kind="bar", ylabel='Number of Passengers', color=colors)
ax.legend(['Not Survived', 'Survived'])
```

```
survival_by_partner
addCaption(survival_by_partner,'df', None, ax)
```



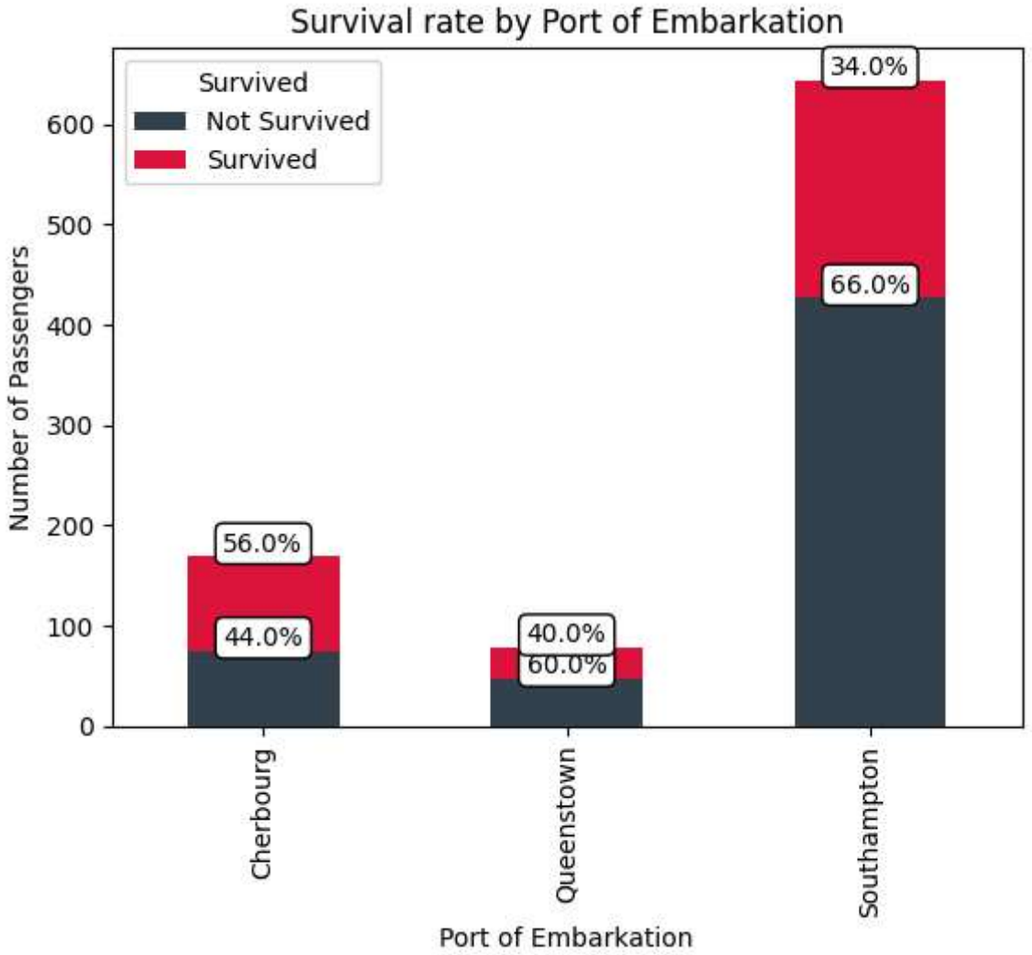
- Most casualties of the shipwreck were passengers who traveled alone, the plot shows 70% of passengers traveling alone did not survive the wreck, while 30% of those in this category survived.
- Out of the passengers traveling with one or more family members, 51% survived the accident, while 49% never survived.

#3d) Analysis of Survival rate by Port of Embarkation

In [ ]:
```
survival_by_port = data.groupby([data['Embarked'],'Survived']).size().unstack()
survival_group = survival_by_port.rename(columns={0: 'Not Survived', 1: 'Survived'})
ax = survival_group.plot(kind="bar", title="Survival rate by Port of Embarkation", stacked=True,\
                    xlabel='Port of Embarkation',ylabel='Number of Passengers', color=colors)
ax.set_xticklabels(['Cherbourg', 'Queenstown', 'Southampton'])
addCaption(survival_by_port,'df', None, ax,)
survival_group
```

Out[ ]:

Survived	Not Survived	Survived
Embarked		
C	75	94
Q	47	31
S	427	217

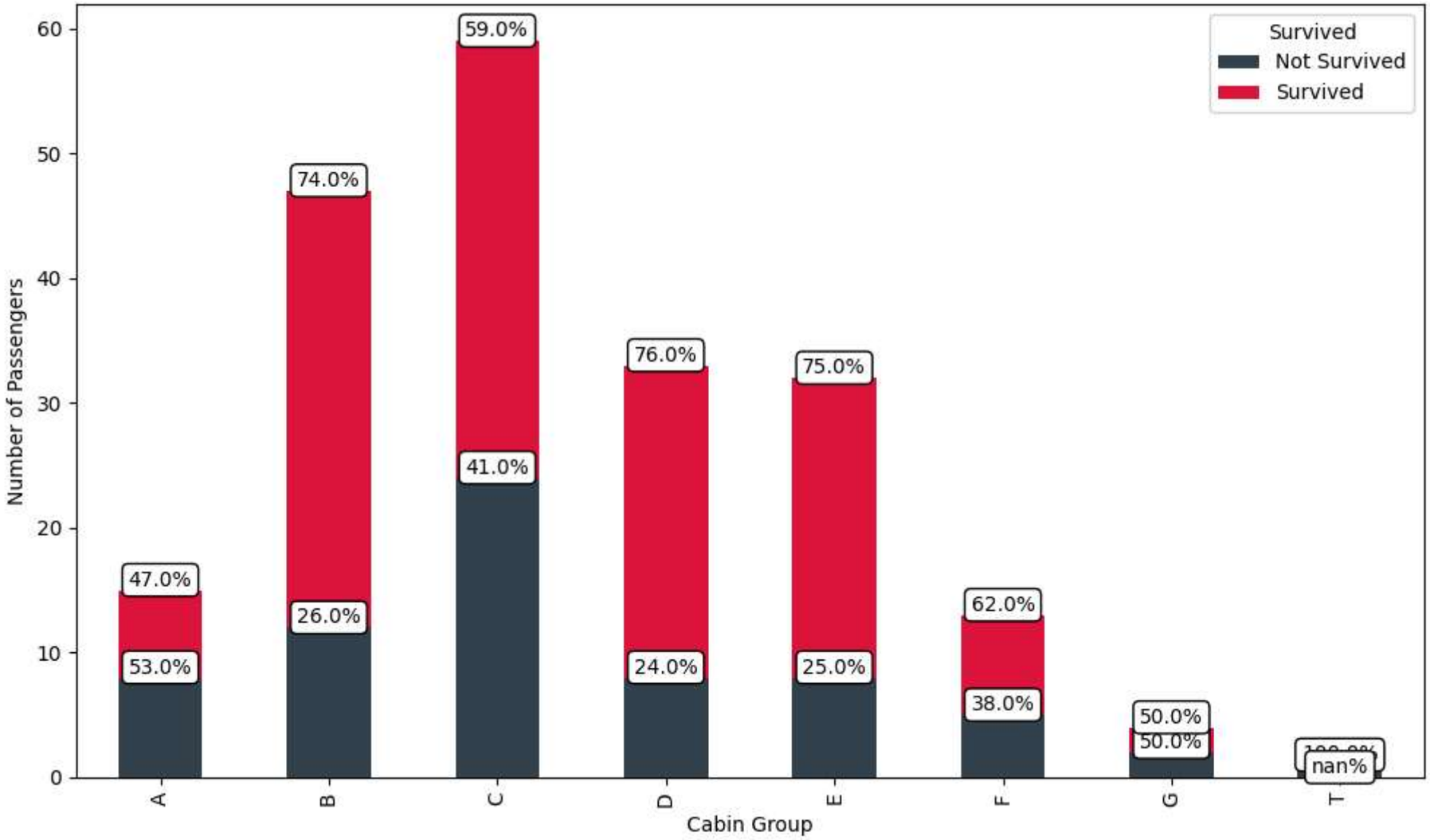


- Most of the casualties are passengers who boarded at the Southampton Port, this is proportional since the port also recorded the highest number of passengers on the ship.
- The Southampton Port recorded 217 at 34% of its passengers as survivors, while 427 at 66% of its passengers never survived.
- The Cherbourg Port recorded 94 at 56% of its passengers as survivors, while 75 at 44% of its passengers never survived.
- The Queenstown Port recorded 31 at 40% of its passengers as survivors, while 47 at 60% of its passengers never survived.

#3e) Analysis of Survival rate by Cabin Group

In [ ]:

```
#fig, axes = plt.subplots(1,2, figsize=(10,6))
grouped_data = raw_data.groupby([raw_data['Cabin'].str[0], 'Survived']).size().unstack()
survival_by_cabin = grouped_data.rename(columns={0:'Not Survived', 1:'Survived'})

ax = survival_by_cabin.plot(kind="bar", figsize=(10, 6), stacked=True, color=colors, \
                        xlabel='Cabin Group', ylabel='Number of Passengers')
addCaption(survival_by_cabin, 'df',None, ax)
plt.tight_layout()
plt.show()

survival_by_cabin
```


Survived
Not Survived
Survived
Number of Passengers
Cabin Group
0
10
20
30
40
50
60
A
B
C
D
E
F
G
T
47.0%
53.0%
74.0%
26.0%
59.0%
41.0%
76.0%
24.0%
75.0%
25.0%
62.0%
38.0%
50.0%
50.0%
nan%

Out[ ]:

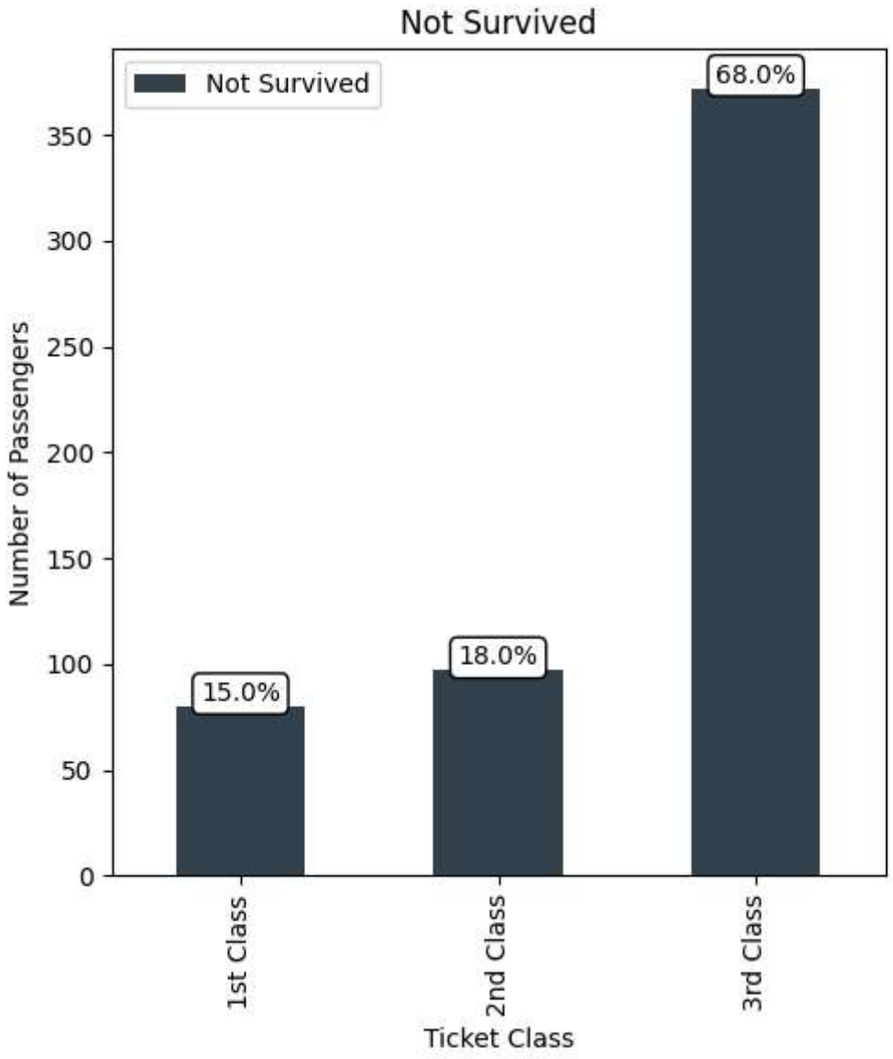
Cabin	Survived	Not Survived	Survived
A	8.0	7.0
B	12.0	35.0
C	24.0	35.0
D	8.0	25.0
E	8.0	24.0
F	5.0	8.0
G	2.0	2.0
T	1.0	NaN

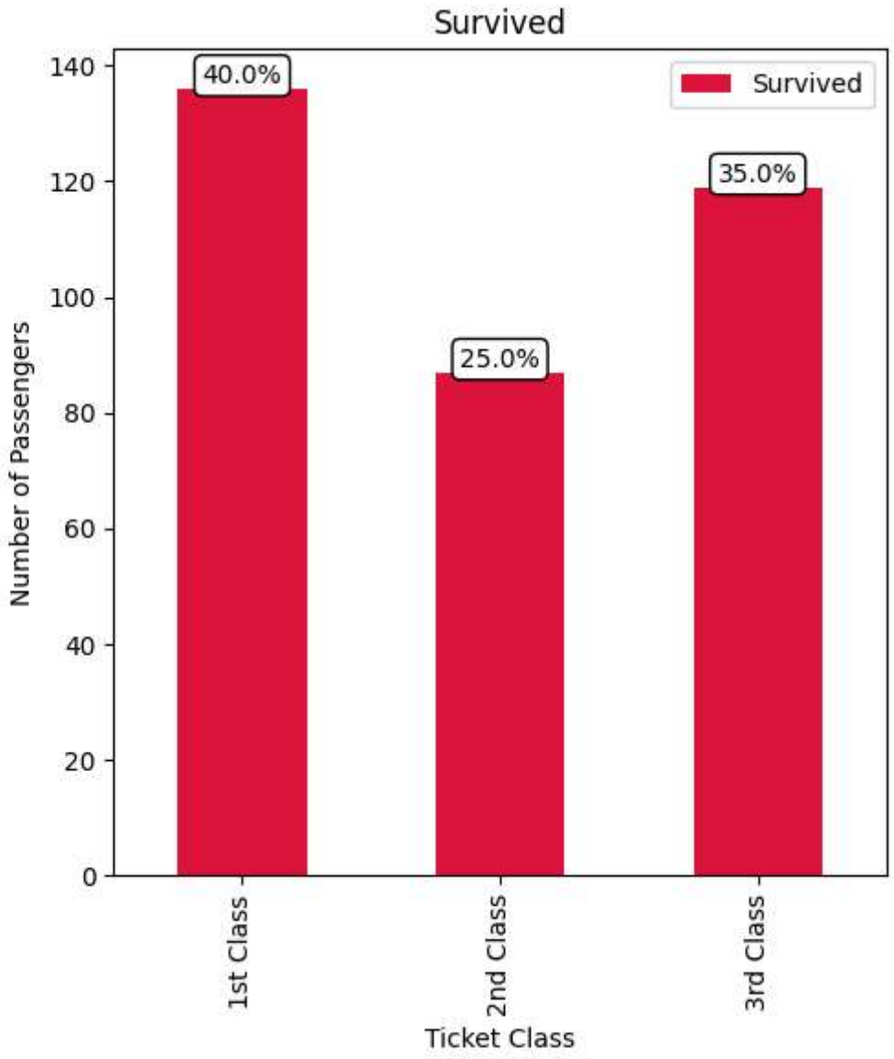
- Out of the 204 Cabin numbers recorded, Cabins C has the highest number of surviving and non-surviving passengers. 35 (59%) passengers survived, while 24 (41%) were non-survivors.

NOTE: The figures above does not represent the true state of the survivors by Cabin, this is because the Cabin column has too many missing values, up to 77% of the entries.

#3f) Analysis of Survival rate by Ticket Class

In [ ]:
```
fig, axes = plt.subplots(1,2, figsize=(10,6))
grouped_ticket = data.groupby([data['Pclass'], 'Survived']).size().unstack()
survival_by_pclass = grouped_ticket.rename(columns={0:'Not Survived', 1:'Survived'})

survival_by_pclass.plot(kind="bar", subplots=True, color=colors, xlabel='Ticket Class', \
                        ylabel='Number of Passengers', ax=axes)
addCaption(survival_by_pclass, 'df', axes, None, ['1st Class', '2nd Class', '3rd Class'])
plt.tight_layout()
plt.show()

survival_by_pclass
```


Not Survived
Not Survived
68.0%
18.0%
15.0%
Number of Passengers
0
50
100
150
200
250
300
350
1st Class
2nd Class
3rd Class
Ticket Class


Survived
Survived
40.0%
25.0%
35.0%
Number of Passengers
0
20
40
60
80
100
120
140
1st Class
2nd Class
3rd Class
Ticket Class

Out[ ]:

Pclass	Survived	Not Survived	Survived
1	80	136	
2	97	87	
3	372	119	

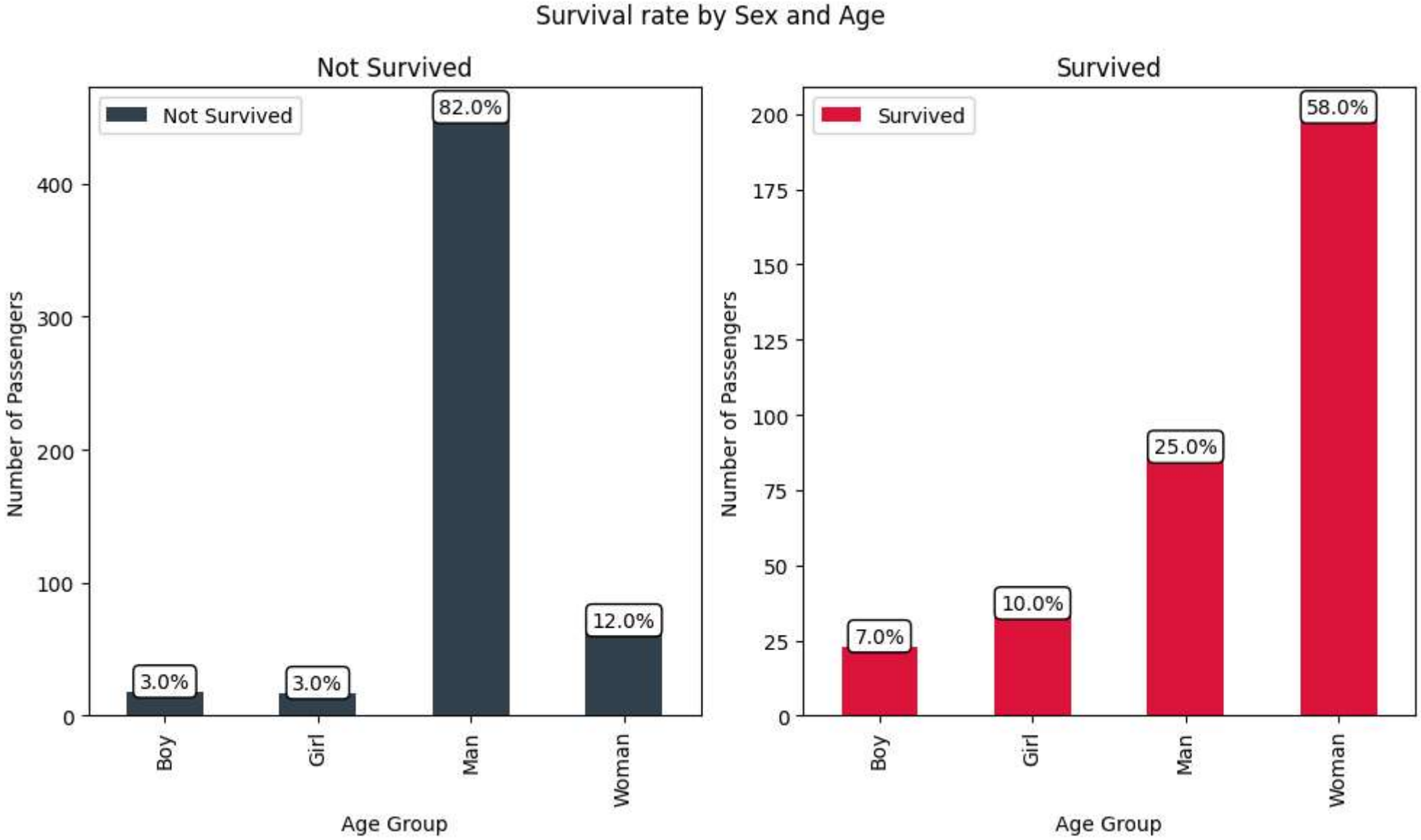
- Passengers with 1st Class tickets formed the highest number of survivors, totalling 136 in number, representing 40% of the total survivors
- Passengers with 3rd Class tickets formed the highest number of survivors, totalling 119 in number, representing 35% of the total survivors
- Passengers with 2nd Class tickets formed the highest number of survivors, totalling 87 in number, representing 25% of the total survivors

Meanwhile, passengers in the 3rd Class made the highest number of non-survivors at a count of 372 at a high 68% of the total non-survivors.

#3g) Analysis of Survival rate by Sex and Age

In [ ]:

```
fig, axes = plt.subplots(1,2, figsize=(10,6))
grouped_age = data.groupby([data['MWBG'], 'Survived']).size().unstack()
survival_by_age = grouped_age.rename(columns={0:'Not Survived', 1:'Survived'})

survival_by_age.plot(kind="bar", subplots=True, color=colors, xlabel='Age Group', \
                     title = 'Survival rate by Sex and Age', ylabel='Number of Passengers', ax=axes)
addCaption(survival_by_age, 'df', axes, None)
plt.tight_layout()
plt.show()

survival_by_age
```

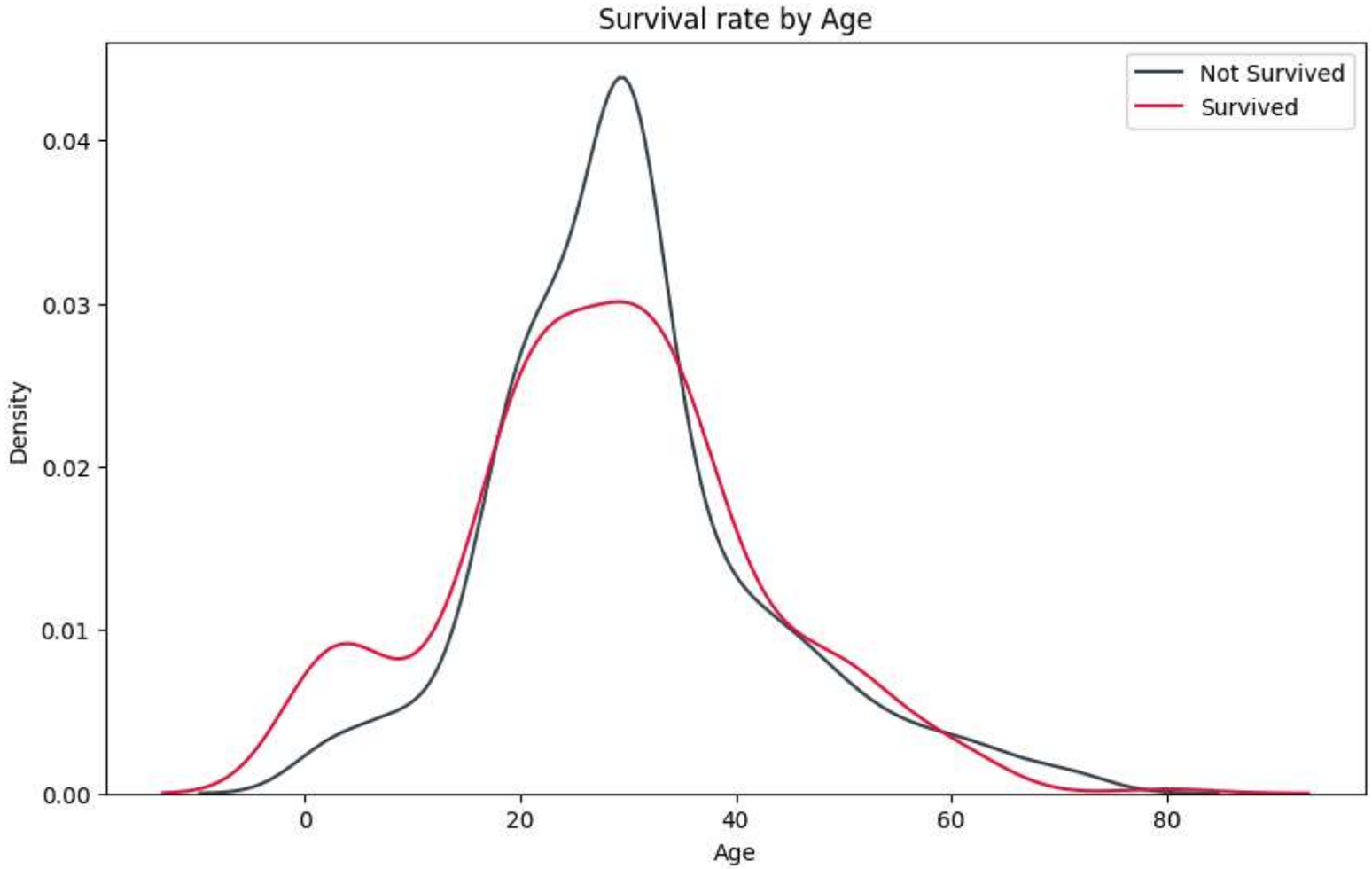

Survival rate by Sex and Age
Not Survived
Not Survived
82.0%
12.0%
3.0%
3.0%
Number of Passengers
0
100
200
300
400
Boy
Girl
Man
Woman
Age Group
Survived
Survived
58.0%
25.0%
10.0%
7.0%
Number of Passengers
0
25
50
75
100
125
150
175
200
Boy
Girl
Man
Woman
Age Group

Out[ ]:

Survived MWBG	Not Survived	Survived
Boy	18	23
Girl	17	34
Man	450	86
Woman	64	199

- Women formed the most survivors of the Titanic shipwreck, forming 58% of the survivors, followed by men at 25% of the survivors, girls formed 10% and boys formed 7% of the overall survivors.
- Men formed the highest number of non-survivors, with a count of 450, a huge 82% of the non-survivors.

In [ ]:
```
plt.figure(figsize=(10, 6))
sns.kdeplot(data[data['Survived'] == 0]['Age'], label='Not Survived', color=colors[0])
sns.kdeplot(data[data['Survived'] == 1]['Age'], label='Survived', color=colors[1])

plt.title('Survival rate by Age')
plt.xlabel('Age')
plt.ylabel('Density')
plt.legend()
plt.show()
```



- The survival rate appears to be higher for younger ages, indicating that age played a significant role in survival at the start of the plot and then slightly fall and the picked up as the age increases towards 20.
- The survival rate appears to peak around early adulthood, suggesting that individuals in this age group had a higher chance of survival. There was a spike in the rate for passengers around age 30, at the same time the risk of death increases with passenger's age. The plot shows a decline in survival rates from around age 30 towards the right side of the plot.