titanic

May 9, 2024

[1]:
```
import pandas as pd
import numpy as np

# plotting modules
import seaborn as sns
import matplotlib.pyplot as plt

# for data encoding and transformation
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer

# for warnings
import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
# df will be used for encoding and scaling
df = pd.read_csv("Titanic.csv")
```

[3]:
```
df.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
```

[4]:
```
df.shape
```

[4]:
```
(891, 12)
```

As we can see above the data set has 12 columns or features and 891 rows. Each row has info about a passenger so in total we have data of 891 passengers. We have numerical and categorical types of features

Columns Description

Passenger Id- Id number of passengers in the data set

Name- Name of the passenger

Survival- Person survived or not (0 for No & 1 for Yes)

P-class- With what class of ticket the passenger was traveling.

sex- Male Or Female

Age- Age of the person in Years

Sibsp- Number of siblings/spouses on the Titanic

parch- Number of parents/children on the Titanic

Ticket- Ticket number

Fare- Amount of money that person paid to travel

Cabin- Cabin Number of Passenger

Embarked- Port of Embarkment (C = Cherbourg, Q = Queenstown, S = Southampton)

[5]:
```
# lets see a few rows for better understanding of dataset
df.head(10)
```

[5]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th…	female	38.0	1
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0
3	4	1	1				
4	5	0	3				
5	6	0	3				
6	7	0	1				
7	8	0	3				
8	9	1	3				
9	10	1	2				

```
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                            Allen, Mr. William Henry    male  35.0      0
5                                    Moran, Mr. James    male   NaN      0
6                             McCarthy, Mr. Timothy J    male  54.0      0
7                      Palsson, Master. Gosta Leonard    male   2.0      3
8  Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)  female  27.0      0
9                  Nasser, Mrs. Nicholas (Adele Achem)  female  14.0      1

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
5      0            330877   8.4583   NaN        Q
6      0             17463  51.8625   E46        S
7      1            349909  21.0750   NaN        S
8      2            347742  11.1333   NaN        S
9      0            237736  30.0708   NaN        C
```

[6]:
```
# Check how many null values are there in each column
df.isnull().sum()
```

[6]:
```
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 the age column we have 177 Null values. In Embarked column we hace 2 missing values. Similarly in the Cabin feature, we have 687 Null values which is Huge. we have only 23% of values present in the data set and 77% of values are missing.

[7]:
```
# Now to check if there are any duplicate values in the dataset.
df.duplicated().sum()
```

[7]:
```
0
```

Since Cabin coulumn has many missing values, we can drop this feature while making our model as it will affect our calculations. Also PassengerId doesnt seem to play any role in the dataset. So we can drop that column too. Name and ticket also dont seem to affect chances of survival so its

best to get rid of them as well so that we can focus on the necessary Features.

[8]:
```
drop_list=['PassengerId','Cabin', 'Name', 'Ticket']
df.drop(drop_list, axis = 1, inplace=True)
```

[9]:
```
df.head(10)
```

[9]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S
1	1	1	female	38.0	1	0	71.2833	C
2	1	3	female	26.0	0	0	7.9250	S
3	1	1	female	35.0	1	0	53.1000	S
4	0	3	male	35.0	0	0	8.0500	S
5	0	3	male	NaN	0	0	8.4583	Q
6	0	1	male	54.0	0	0	51.8625	S
7	0	3	male	2.0	3	1	21.0750	S
8	1	3	female	27.0	0	2	11.1333	S
9	1	2	female	14.0	1	0	30.0708	C

[10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  891 non-null    int64
 1   Pclass    891 non-null    int64
 2   Sex       891 non-null    object
 3   Age       714 non-null    float64
 4   SibSp     891 non-null    int64
 5   Parch     891 non-null    int64
 6   Fare      891 non-null    float64
 7   Embarked  889 non-null    object
dtypes: float64(2), int64(4), object(2)
memory usage: 55.8+ KB
```

[11]:
```
df.describe()
```

[11]:

	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

Now to deal with missing Age values

Average age in the dataset is 29.7 years with std deviation of 14.52. It means most of the people have age in the range (29.7 - 14.52) to (29.7 + 14.52) because in a Continuous Random Variable most of the values can be found in the range of (mean-std) to (mean+std). The min age is 0.42 years and max age is 80 years old. So I am going to fill in the missing age with mean age which is 29.7= 30 here.

[12]:
```
df.Age.fillna(30, inplace=True)
```

[13]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Survived  891 non-null    int64
 1   Pclass    891 non-null    int64
 2   Sex       891 non-null    object
 3   Age       891 non-null    float64
 4   SibSp     891 non-null    int64
 5   Parch     891 non-null    int64
 6   Fare      891 non-null    float64
 7   Embarked  889 non-null    object
dtypes: float64(2), int64(4), object(2)
memory usage: 55.8+ KB
```

Now to deal with 2 missing Embarked values For 2 missing values of Embarked, I am going to fill in those values with most frequent value since Embarked is a Category datatype. For that I will use value_counts to find out the most frequent value in Embraked column.

[14]:
```
df.Embarked.value_counts()
```

[14]:
```
Embarked
S    644
C    168
Q     77
Name: count, dtype: int64
```

[15]:
```
#df.Embarked.replace('nan', 'S', inplace=True)
# Since S is the most most frequent, I will fill in the missing values with S.
df.Embarked.fillna('S', inplace=True)
```

[16]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 8 columns):
 #   Column    Non-Null Count  Dtype
```

```
---  ------    --------------  -----
 0   Survived  891 non-null    int64
 1   Pclass    891 non-null    int64
 2   Sex       891 non-null    object
 3   Age       891 non-null    float64
 4   SibSp     891 non-null    int64
 5   Parch     891 non-null    int64
 6   Fare      891 non-null    float64
 7   Embarked  891 non-null    object
dtypes: float64(2), int64(4), object(2)
memory usage: 55.8+ KB
```

Sex and Embarked are both object type, nominal features, we need to encode them to get the best EDA from our data.

[17]:
```
nominal_features= ['Sex', 'Embarked']
transformer= make_column_transformer((OneHotEncoder(), df.
 ↪select_dtypes('object').columns.tolist()),
                                     verbose_feature_names_out=False)

data_encoded=transformer.fit_transform(df)
data=pd.DataFrame(data_encoded, columns= transformer.get_feature_names_out())

# Drop Original nominal columns

df.drop(columns=nominal_features, axis=1, inplace=True)
df = pd.concat([df, data], axis=1)
```

[18]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Survived    891 non-null    int64
 1   Pclass      891 non-null    int64
 2   Age         891 non-null    float64
 3   SibSp       891 non-null    int64
 4   Parch       891 non-null    int64
 5   Fare        891 non-null    float64
 6   Sex_female  891 non-null    float64
 7   Sex_male    891 non-null    float64
 8   Embarked_C  891 non-null    float64
 9   Embarked_Q  891 non-null    float64
 10  Embarked_S  891 non-null    float64
dtypes: float64(7), int64(4)
memory usage: 76.7 KB
```

[19]:
```
df.head(10)
```

[19]:

	Survived	Pclass	Age	SibSp	Parch	Fare	Sex_female	Sex_male	\
0	0	3	22.0	1	0	7.2500	0.0	1.0	
1	1	1	38.0	1	0	71.2833	1.0	0.0	
2	1	3	26.0	0	0	7.9250	1.0	0.0	
3	1	1	35.0	1	0	53.1000	1.0	0.0	
4	0	3	35.0	0	0	8.0500	0.0	1.0	
5	0	3	30.0	0	0	8.4583	0.0	1.0	
6	0	1	54.0	0	0	51.8625	0.0	1.0	
7	0	3	2.0	3	1	21.0750	0.0	1.0	
8	1	3	27.0	0	2	11.1333	1.0	0.0	
9	1	2	14.0	1	0	30.0708	1.0	0.0	

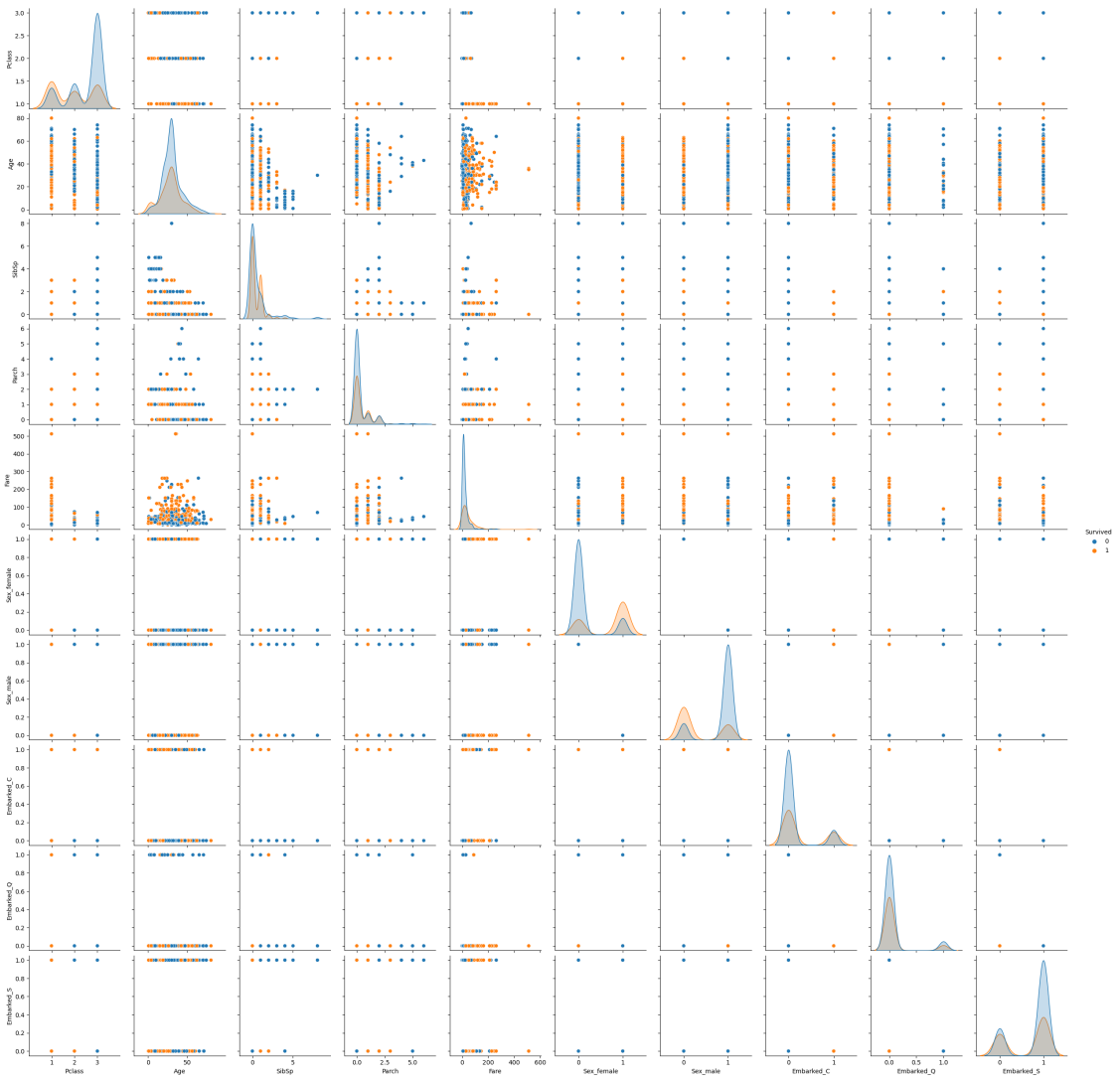
	Embarked_C	Embarked_Q	Embarked_S
0	0.0	0.0	1.0
1	1.0	0.0	0.0
2	0.0	0.0	1.0
3	0.0	0.0	1.0
4	0.0	0.0	1.0
5	0.0	1.0	0.0
6	0.0	0.0	1.0
7	0.0	0.0	1.0
8	0.0	0.0	1.0
9	1.0	0.0	0.0

We have gotten rid of unimportant columns, missing data and have encoded the nominal variables. Next step now would be to focus on the Survived column, which is the target variable or class label here since we want to predict who survived and why?

In order to answer all the questions asked in the assignment, I am going to analyse each column separately and in relation to each other.

[20]:
```
# Plot pairplot first
sns.pairplot(df, hue="Survived")
```

[20]:
```
<seaborn.axisgrid.PairGrid at 0x12f9a1d50>
```

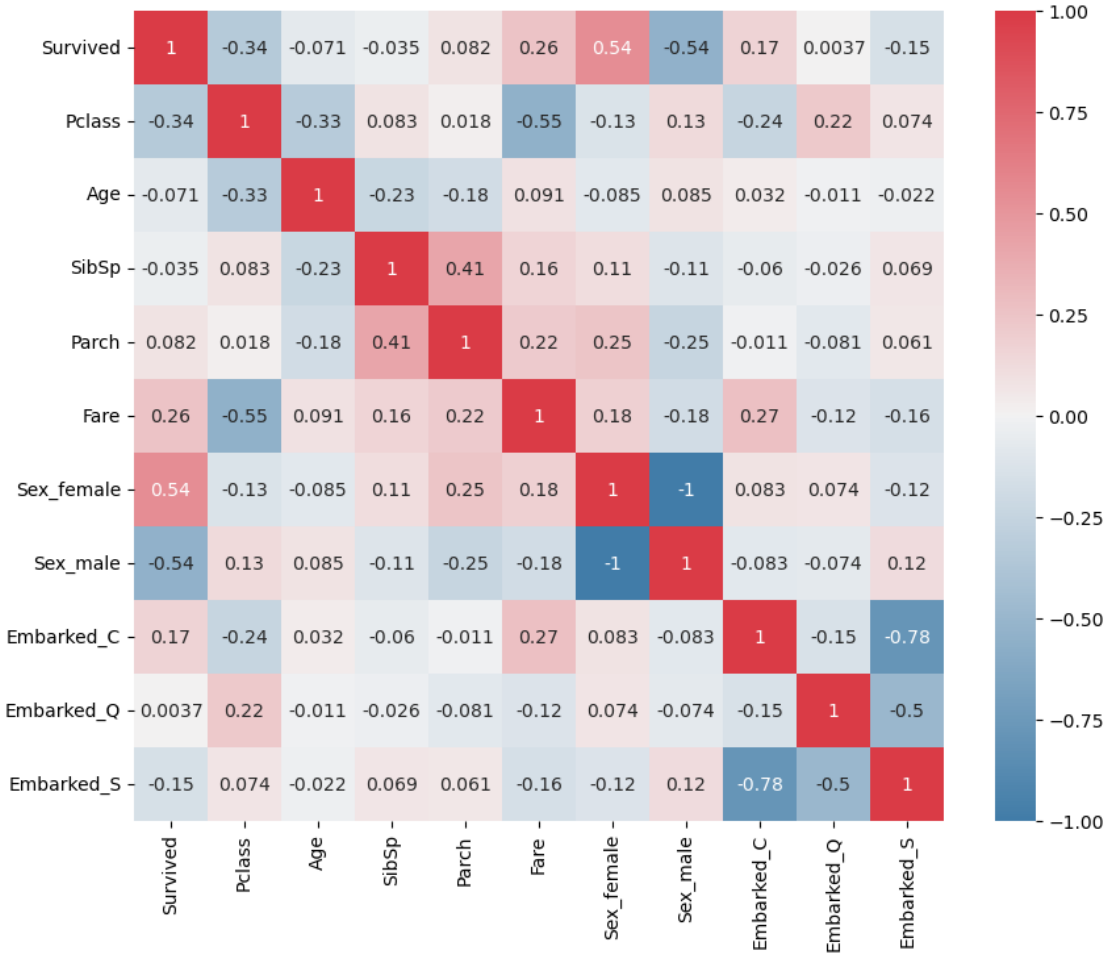


In Survived column, 1 means the person survived and 0 means that person didnt survive. From the above grid of plots, we can see that passengers who paid higher fares or traveling in the upper class had a higher chance to survive. Age is also giving some info that younger persons had a higher chance to survive than older people. Also from the look of it, female had a better chance of survival than the male. The passengers who embarked from Cherbourg had a better chance of survival then the passengers who embraked from Queenstown and Southampton. The picture is not very clear with pair plots so let's go towards heatmap analysis to understand what exactly is happening.

[21]:
```
plt.figure(figsize = (15,5))
f, ax = plt.subplots(figsize=(10, 8))
corr = df.corr()
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool_), cmap=sns.
  ↪diverging_palette(240,10,as_cmap=True),
              square=True, ax=ax, annot=True)
```

[21]: <Axes: >

<Figure size 1500x500 with 0 Axes>

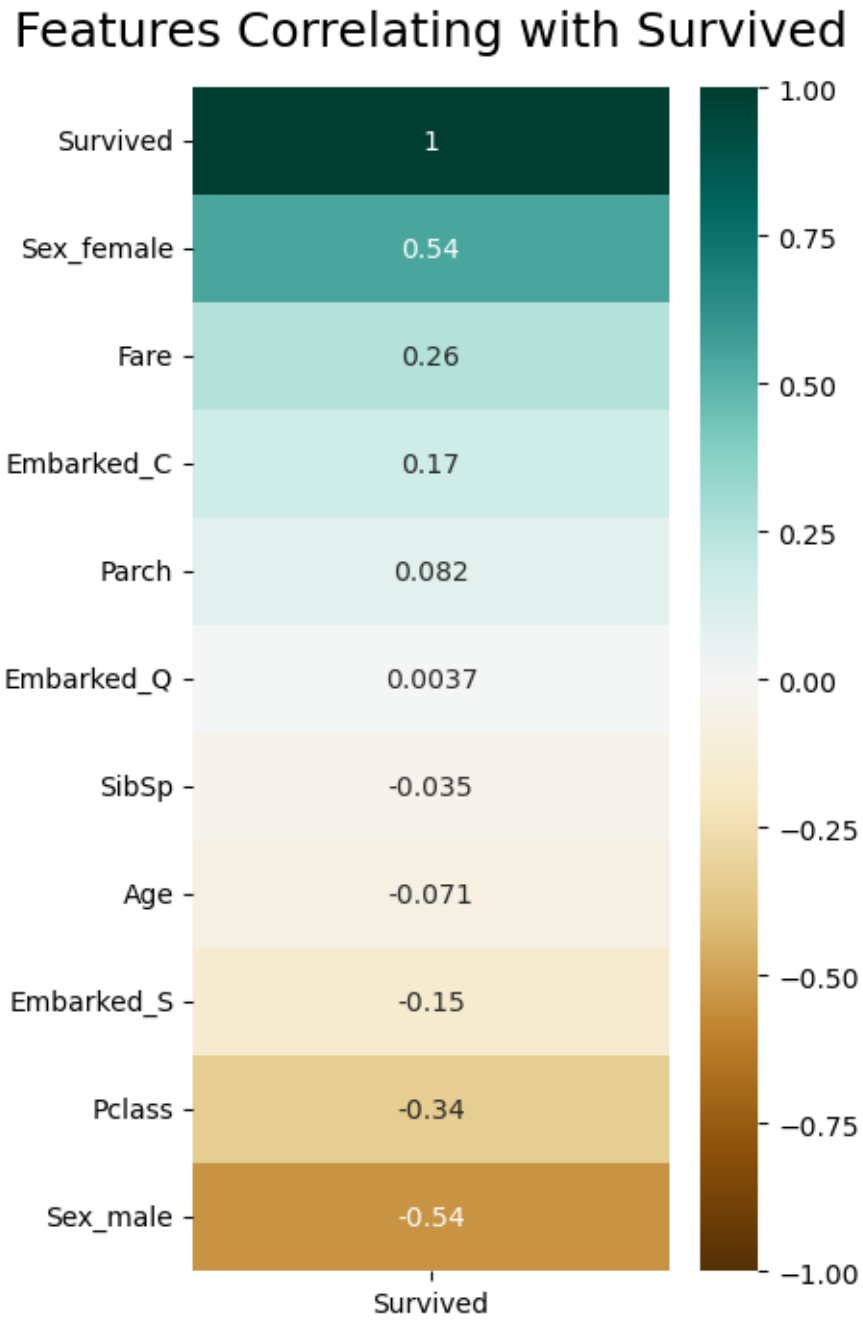


[22]:
```
df.corr()['Survived'].sort_values(ascending= False)
```

[22]:
```
Survived      1.000000
Sex_female    0.543351
Fare          0.257307
Embarked_C    0.168240
Parch         0.081629
Embarked_Q    0.003650
SibSp        -0.035322
Age          -0.070657
Embarked_S   -0.149683
Pclass       -0.338481
Sex_male     -0.543351
```

```
Name: Survived, dtype: float64
```

[23]:
```
plt.figure(figsize=(4, 8))
heatmap = sns.heatmap(df.corr(numeric_only=True)[['Survived']].
 ↪sort_values(by='Survived', ascending=False), vmin=-1, vmax=1, annot=True,␣
 ↪cmap='BrBG')
heatmap.set_title('Features Correlating with Survived', fontdict={'fontsize':
 ↪18}, pad=16);
```



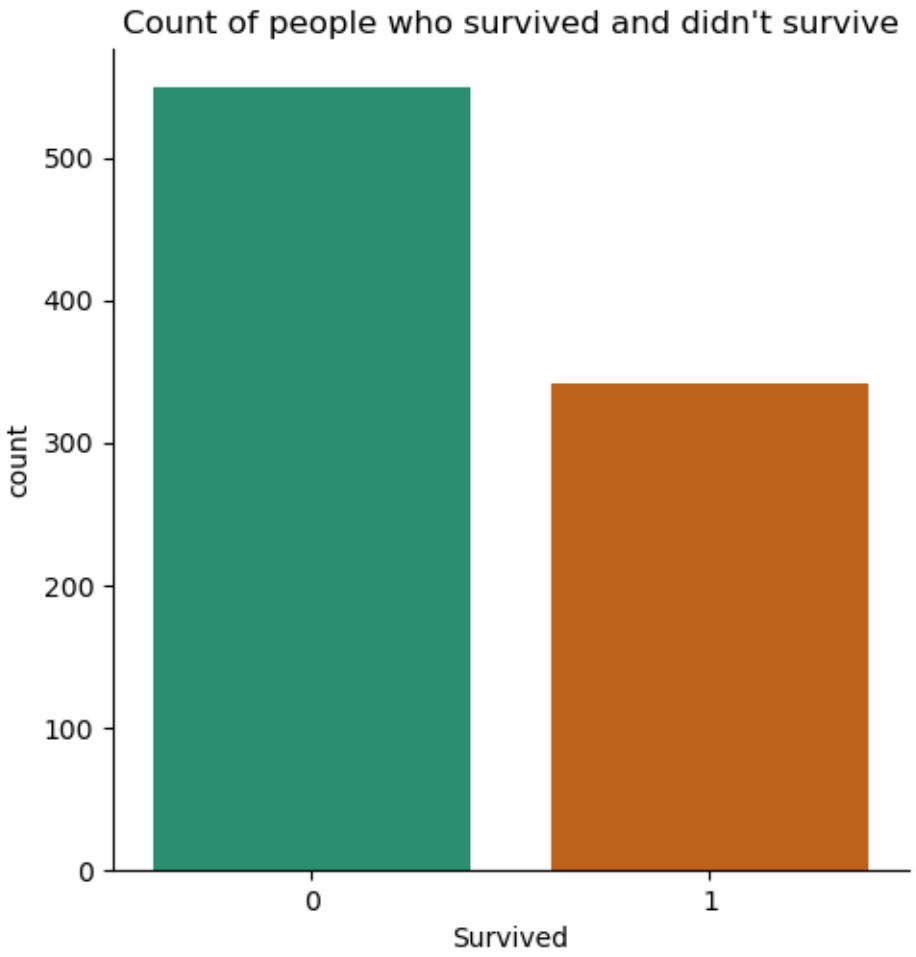
Detailed Analysis based on the Heatmap:

From the above heatmap, we can clearly see that Sex female, Fare, Embarked from Cherbourg, Parch, Embarked from Queendtwon all have positive collinearity with survived (in descending order), that is they are directly proportional to the Survived column. When either of them increases, chances of survival also increase. But SibSp, Age, Embarked_S, Pclass, Sex Male are all inversely proportional to Survived. That is when either of them increases, chances of survival decrease. For Age column it means younger passengers had a better chance of survival. For male column it means Female had a better chance of survival than male. For Sibsp column it means people who had a sibling or spouse onboard with them, had a slimmer chance of survival. Also people who embarked the ship from Southampton survived lesser than the people who embraked the ship from the other two towns. For Pclass it means (1 being the highest or upper class) people in class 3 had the least chance of survival.

That means people in class 1 (upper class), women, younger kids, people who paid more fare, people who embarked from Cherbourg, people with parents or children (Parch) were given priority on life boats.

[24]:
```
# Now to check how many passengers actually survived and how many didnt. I am␣
 ↪using a Countplot, that counts the
# each category of value and plots that.

sns.catplot(x= 'Survived' , kind="count", palette="Dark2", data=df)
plt.title("Count of people who survived and didn't survive")
print(df.Survived.value_counts())
```

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

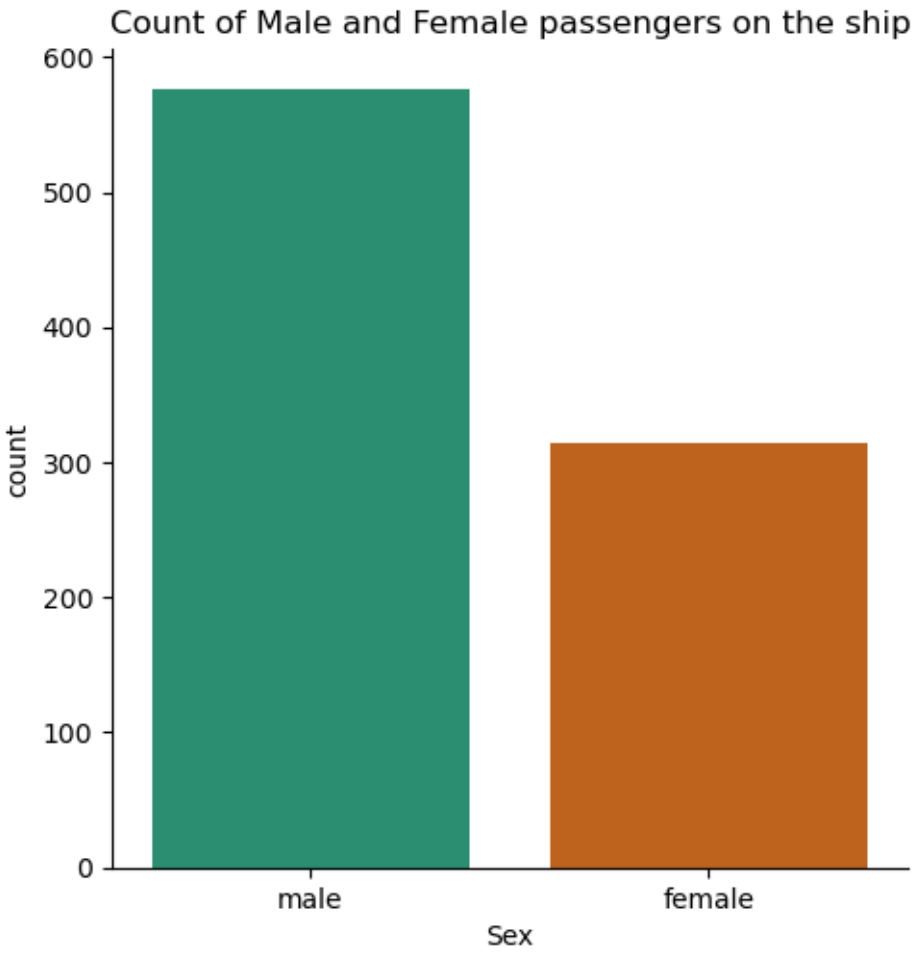


[25]:
```
# data will be used for plotting
data= pd.read_csv("Titanic.csv")
```

In Survived column 1 means the person survived and 0 means that person didnt survive. The analysis shows majority of the people didnt survive, that is out of the total of 891 passengers, 549 didnt survive and 342 passengers survived.

[26]:
```
# Sex column investigation in depth:
sns.catplot(x="Sex", kind="count", palette="Dark2", data=data)
plt.title("Count of Male and Female passengers on the ship")
print(data.Sex.value_counts())
```

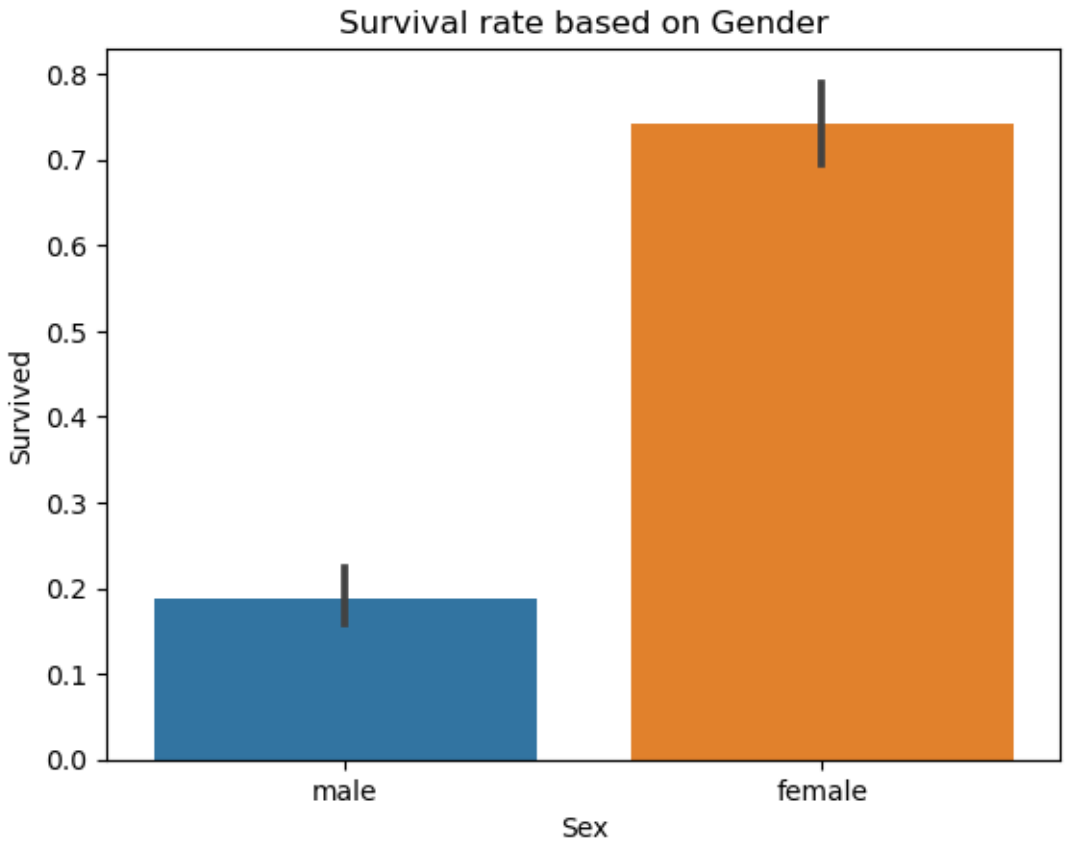
```
Sex
male      577
female    314
Name: count, dtype: int64
```



The analysis shows there were more male passengers on the ship than female.

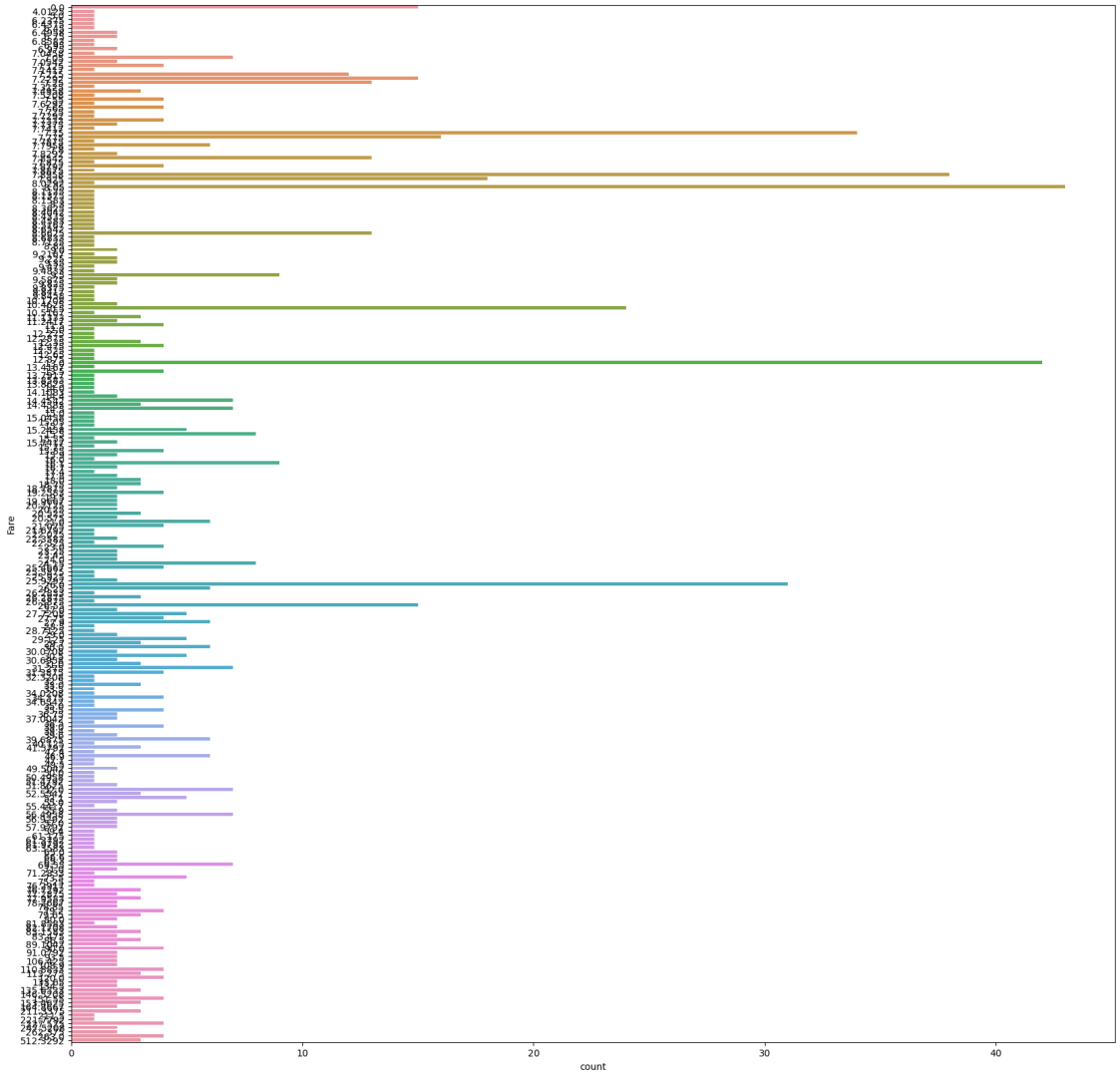
[27]:
```
sns.barplot(data=data, x='Sex', y='Survived')
plt.title("Survival rate based on Gender")
```

[27]:
```
Text(0.5, 1.0, 'Survival rate based on Gender')
```



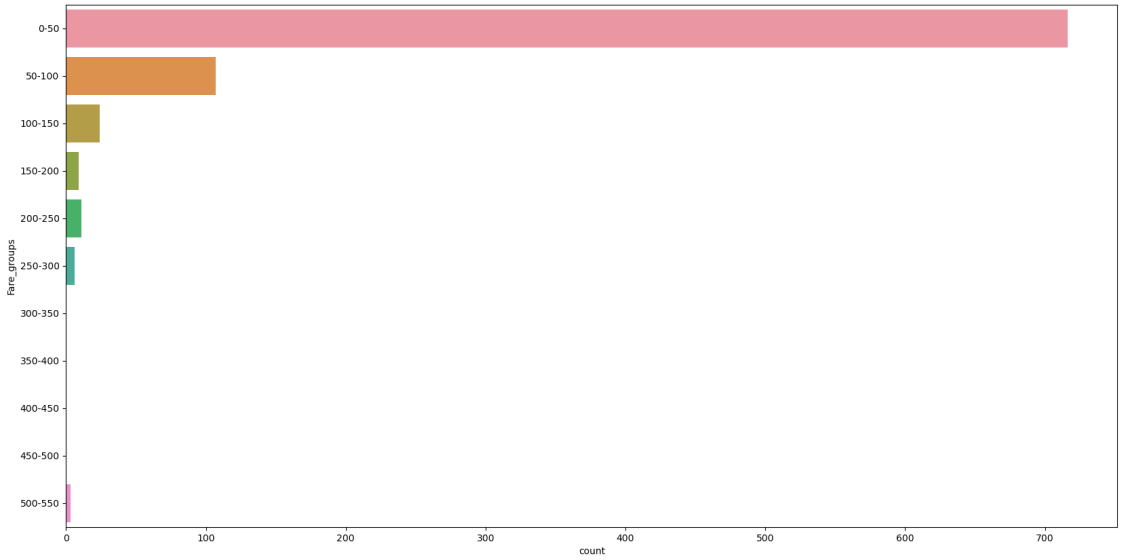
Above barplot further confirms our investigation that more female survived than male.

[28]:
```
# Fare column investigation in depth:
fig = plt.figure(figsize=(20,20))
sns.countplot(y=data['Fare'], data=data);
```



These are so many categories to analyse. I am now going to put them in groups for ease of understanding. Since we are dealing with Fare, I will put the pasengers into groups based on how much Fare they paid. We can do this by putting them into Fare groups of 50. From the describe method we know that the cheapest ticket was for 0 and the most expensive ticket was for $512. So I will start with Fare group 0-50, 50-100 and so on.

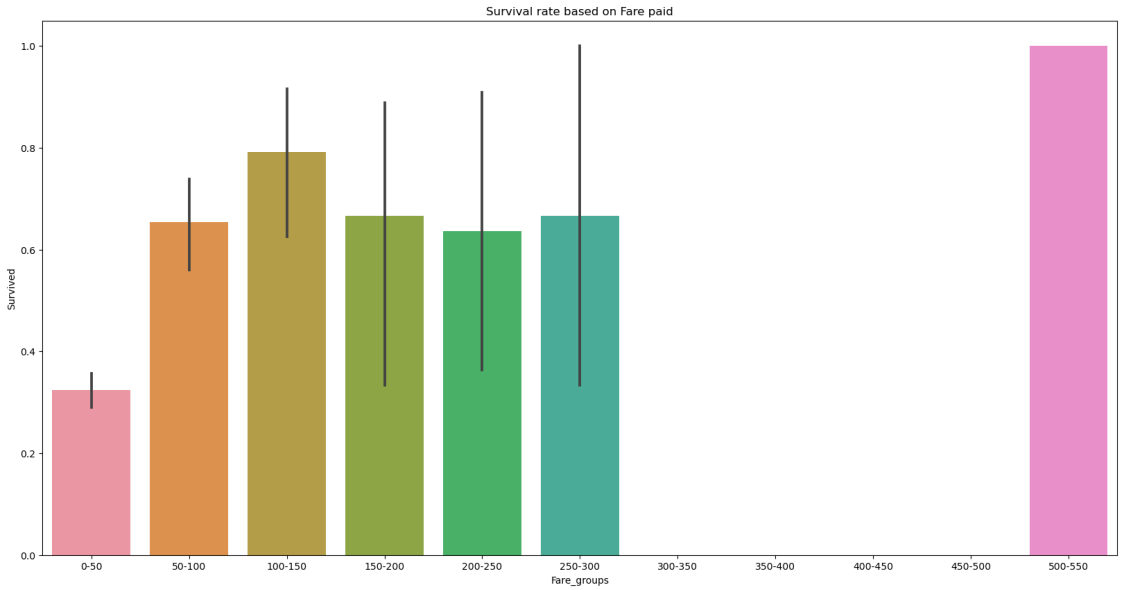
[29]:
```
data['Fare_groups'] = pd.cut(data['Fare'],␣
 ↪[0,50,100,150,200,250,300,350,400,450, 500, 550],
                             ␣
 ↪labels=['0-50','50-100','100-150','150-200','200-250','250-300',
                           '300-350','350-400','400-450','450-500','500-550'])

fig = plt.figure(figsize=(20,10))
sns.countplot(y=data['Fare_groups'], data=data);
```



The above analysis shows majority of the passengers got the cheapest tickets and least number of people got the most expensive tickets.

[30]:
```
# Now that we have grouped passengers based on how much fare they paid we can␣
  ↪see impact of Fare on survival rate.
fig = plt.figure(figsize=(20,10))
sns.barplot(data=data, x='Fare_groups', y='Survived')
plt.title("Survival rate based on Fare paid")
```

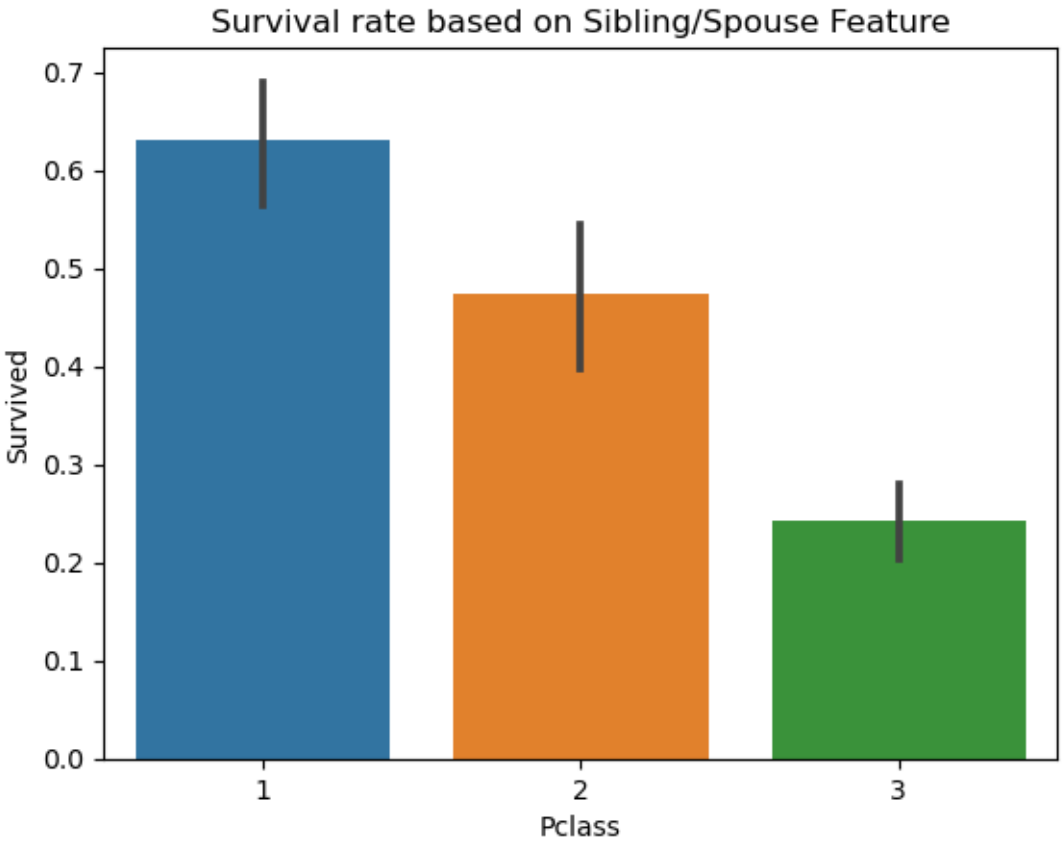
[30]:
```
Text(0.5, 1.0, 'Survival rate based on Fare paid')
```



Again as we have established before, passengers who paid highest fare ($500-$550) survived the most (they were given preference on life boats) and the passengers who paid the lowest fare (0-$50) survived the least.

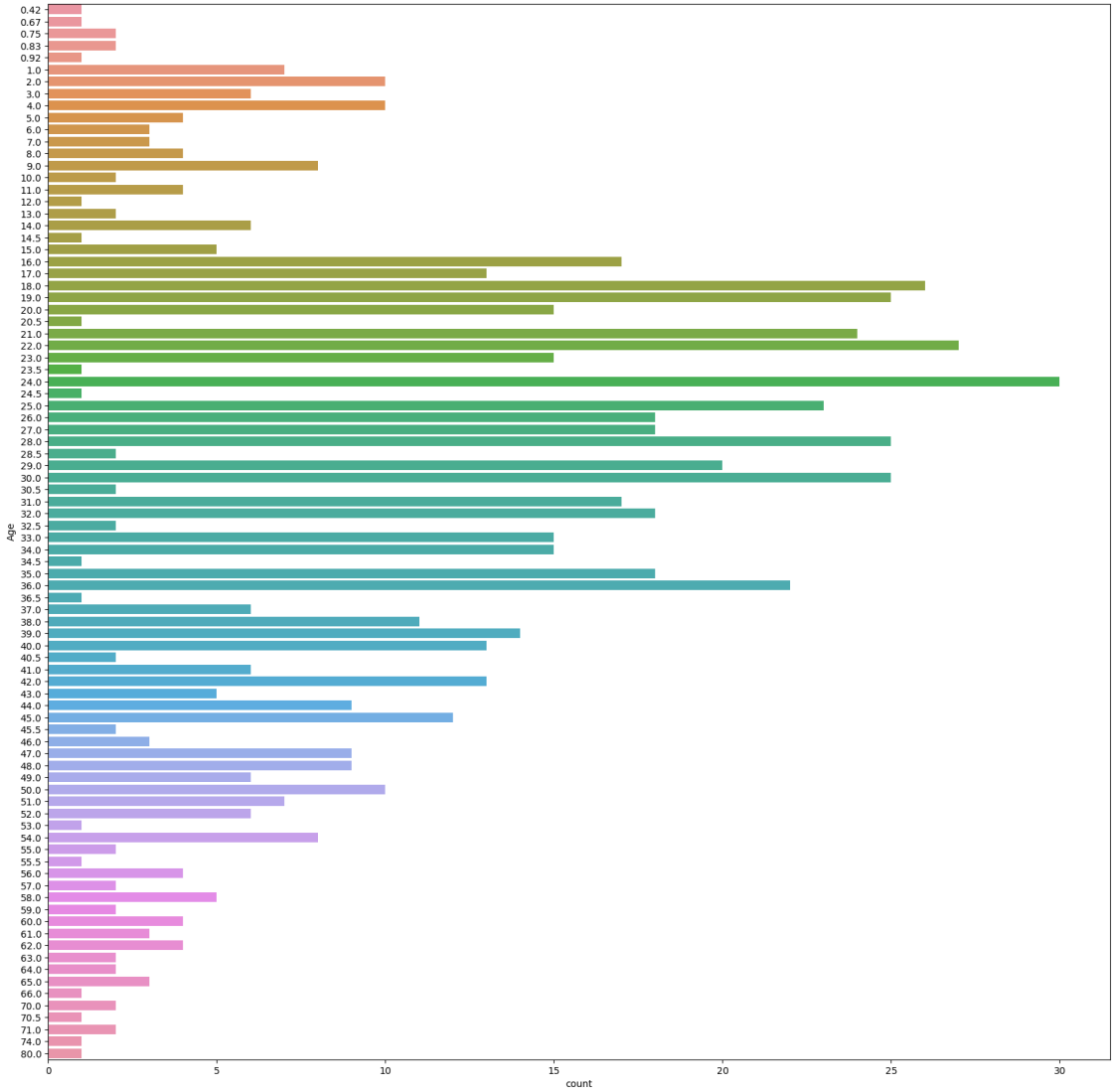
[31]:
```
# Pclass column investigation in depth:
sns.barplot(data=data, x='Pclass', y='Survived')
plt.title("Survival rate based on Sibling/Spouse Feature")
```

[31]:
```
Text(0.5, 1.0, 'Survival rate based on Sibling/Spouse Feature')
```



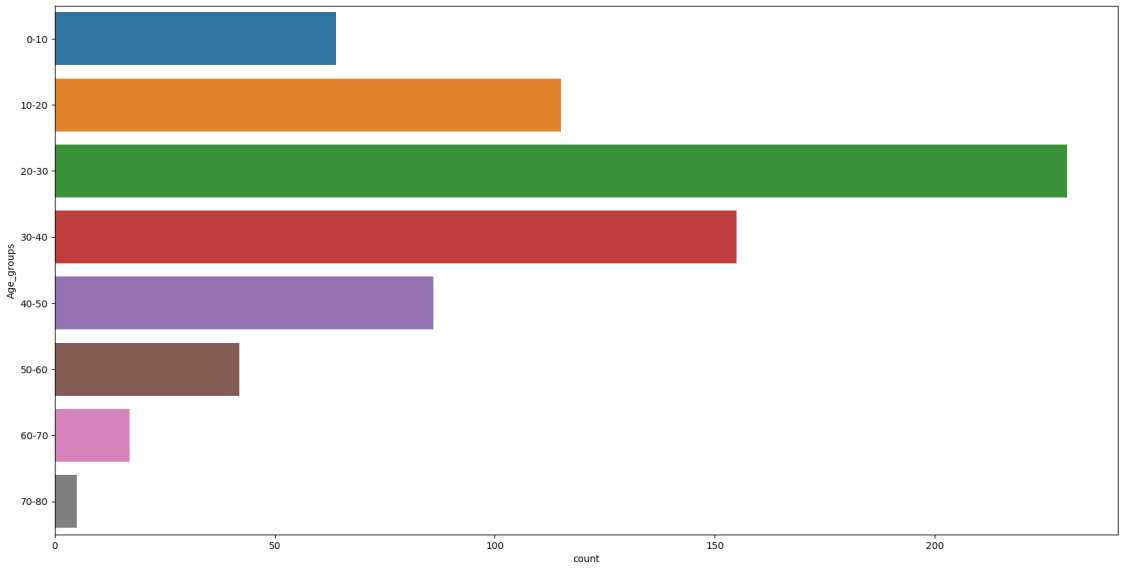
Just like Fare column, in Pclass column if a passenger got ticket of an upper class compartment that is first class donated by 1 in the dataset, his chances of survival increased. Passenegers in class 3 survived the least.

[32]:
```
# Age column investigation in depth:
fig = plt.figure(figsize=(20,20))
sns.countplot(y=data['Age'], data=data);
```



Just like Fare column there are many entries in Age column and its best to make groups for Age column too for better understanding of data and for detecting any pattern. From describe we know that min age is 0.42 years and max is 80 years. Lets divide them into 8 subgroups with age groups 0-10, 10-20 and so on in each subgroup.

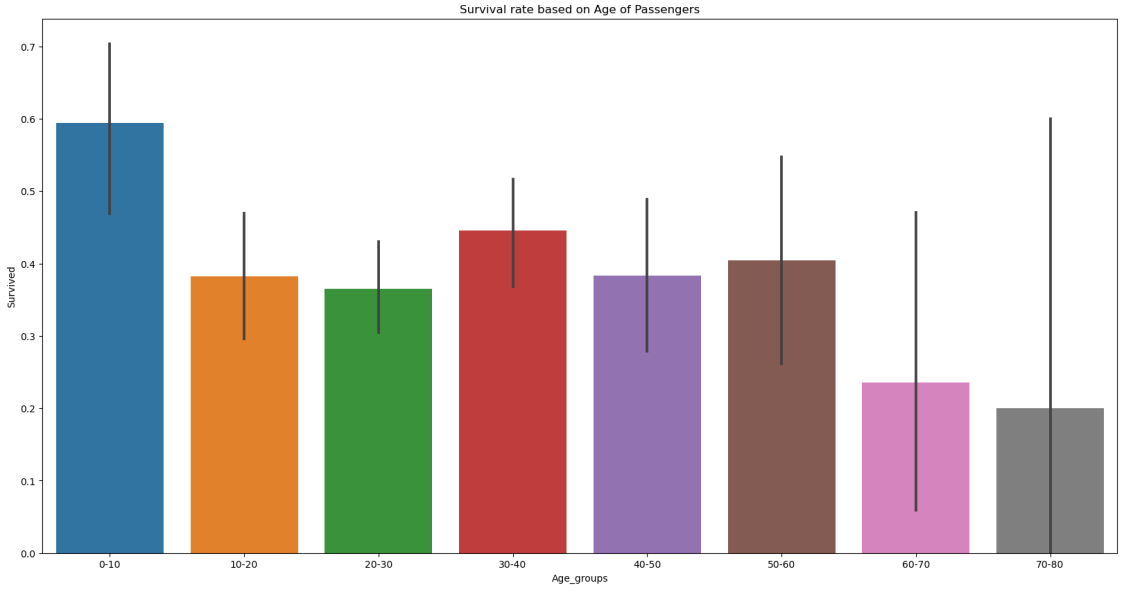
[33]:
```
data['Age_groups'] = pd.cut(data['Age'], [0,10,20,30,40,50,60,70,80],
                    labels=['0-10','10-20','20-30','30-40','40-50','50-60',
                            '60-70','70-80'])

fig = plt.figure(figsize=(20,10))
sns.countplot(y=data['Age_groups'], data=data);
```



The above analysis shows majority of the passengers were from the age group 20-30 and least number of passengers were from the age bracket 70-80.

[34]:
```
# Now that we have grouped passengers based on their Age we can see impact of␣
↪Age on survival rate.
fig = plt.figure(figsize=(20,10))
sns.barplot(data=data, x='Age_groups', y='Survived')
plt.title("Survival rate based on Age of Passengers")
```

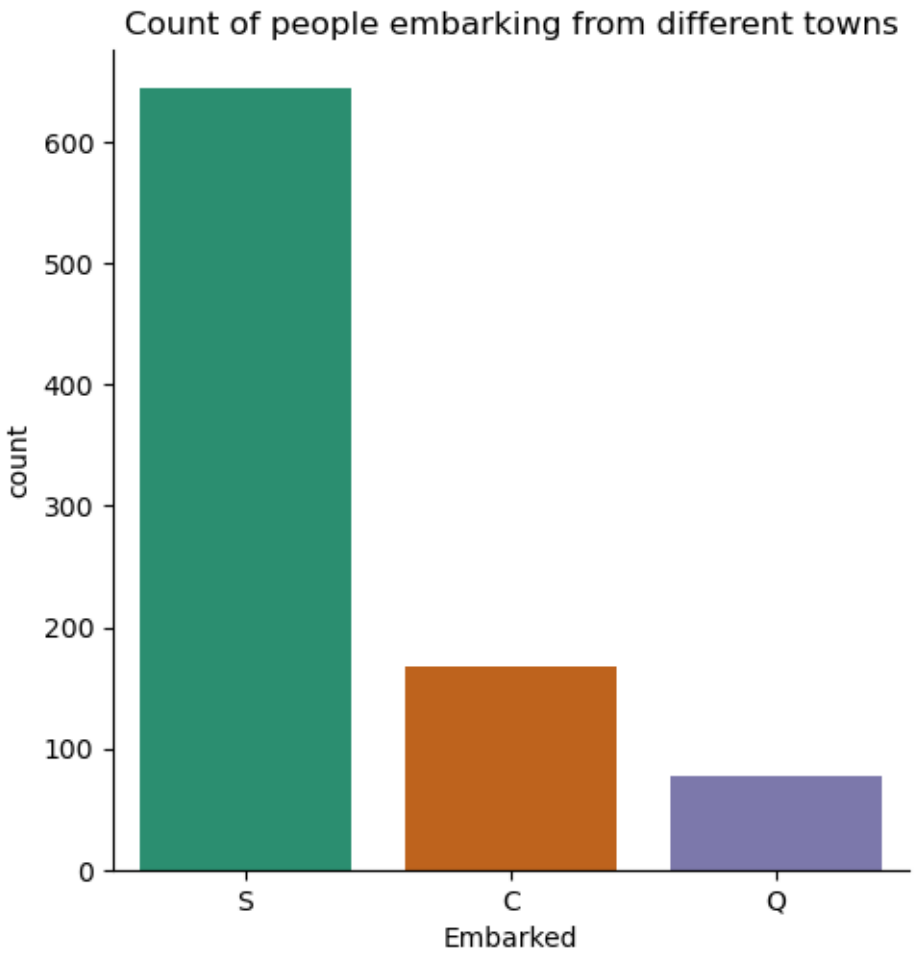
[34]:
```
Text(0.5, 1.0, 'Survival rate based on Age of Passengers')
```



Again as we have established before, youngest passengers (0-10) survival rate was the highest and oldest passengers (70-80) survived the least meaning thereby children were given priority on life boats.

[35]:
```
# Embarked column investigation in depth:
sns.catplot(x="Embarked", kind="count", palette="Dark2", data=data)
plt.title("Count of people embarking from different towns")
```

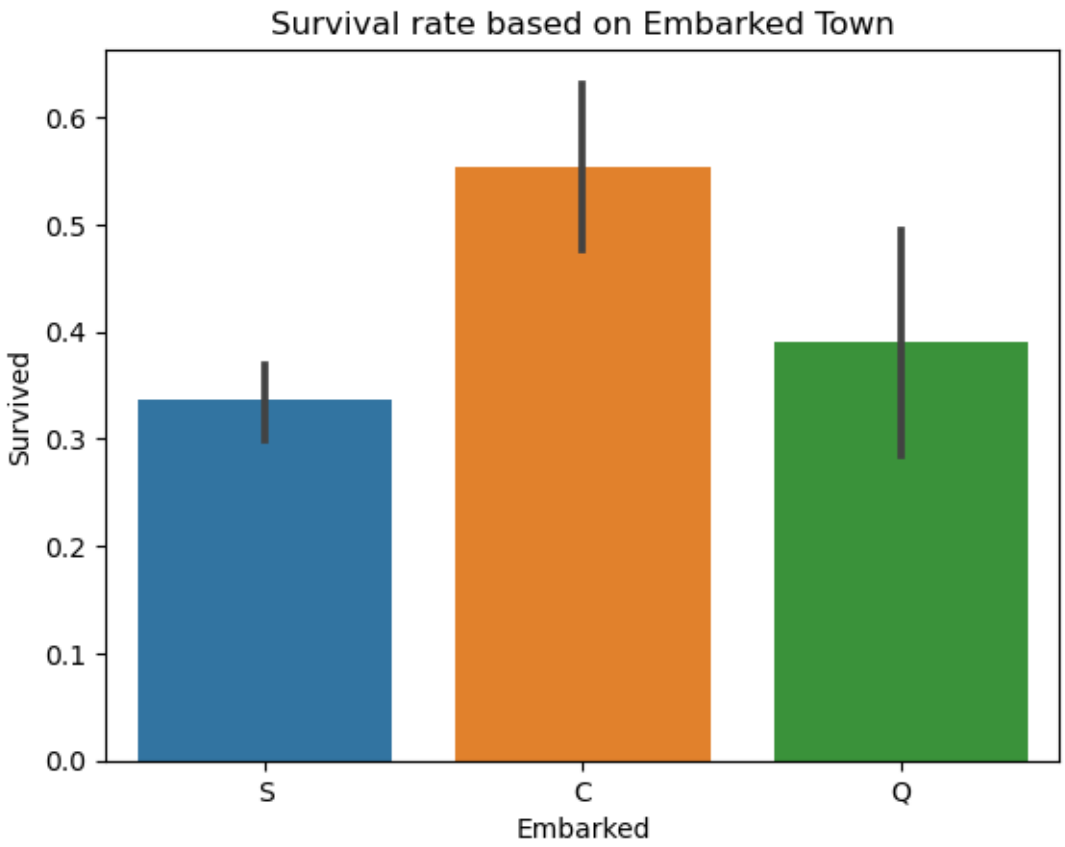
[35]:
```
Text(0.5, 1.0, 'Count of people embarking from different towns')
```



Maximum number of passenegrs embarked from Southampton town and least numb er of people embarked from Queenstown.

[36]:
```
sns.barplot(data=data, x='Embarked', y='Survived')
plt.title("Survival rate based on Embarked Town")
```

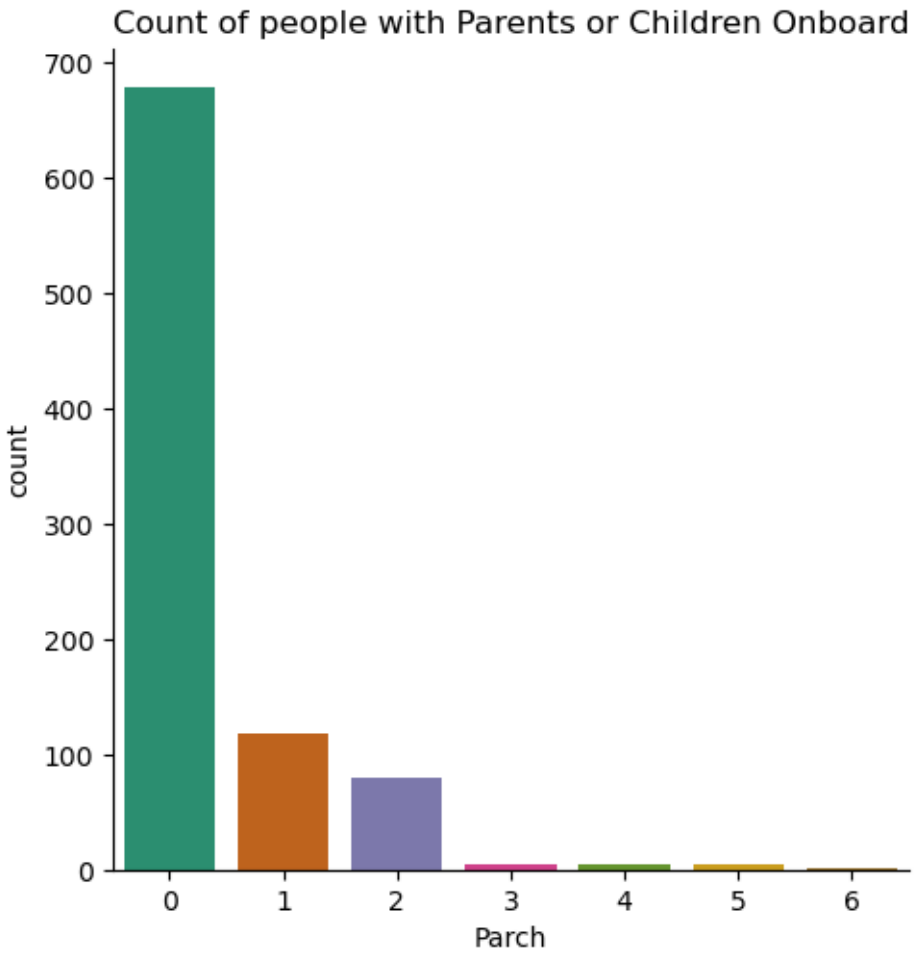
[36]: Text(0.5, 1.0, 'Survival rate based on Embarked Town')



Passengers who embraked from Cherbourg survived the most and passengers who embarked from Southampton town survived the least. Since we already know from Fare column analysis that majority of the people bought cheapest tickets so we can establish the fact that people who embarked the ship from Southampton paid the least Fare and therefore survived the least.

[37]:
```
# Parch column investigation in depth:
sns.catplot(x="Parch", kind="count", palette="Dark2", data=data)
plt.title("Count of people with Parents or Children Onboard")
```

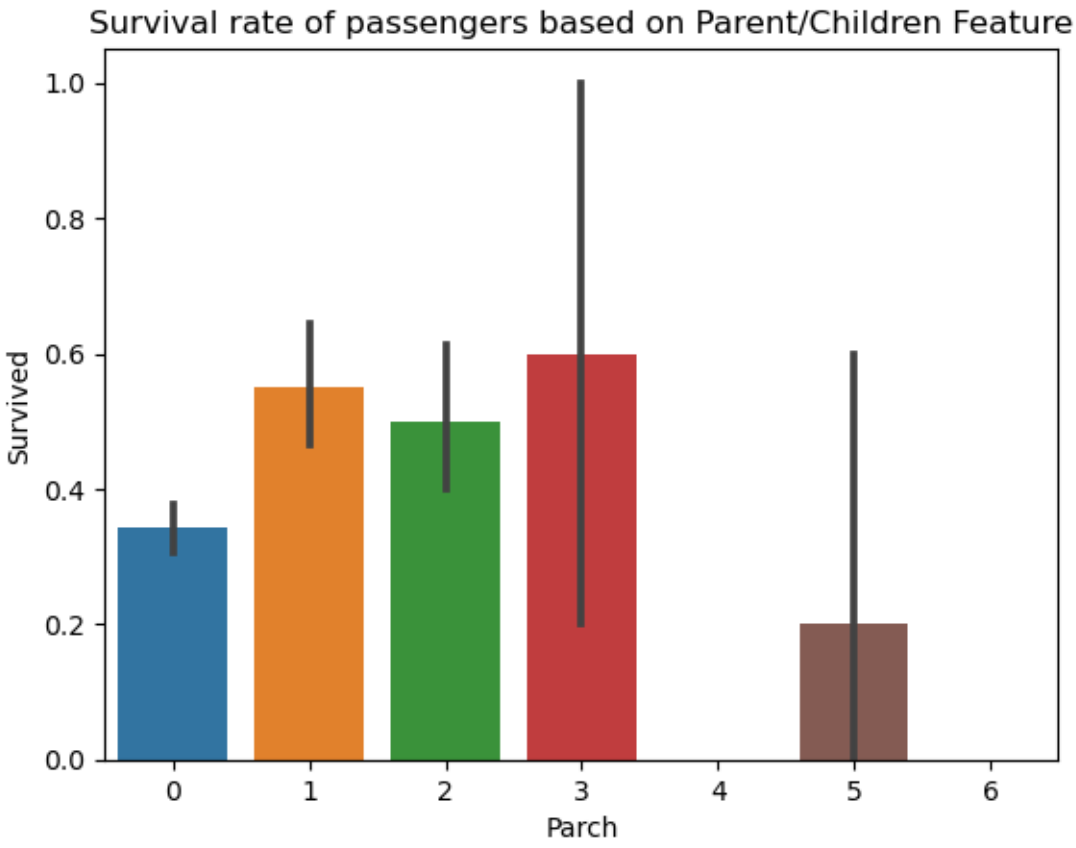
[37]: Text(0.5, 1.0, 'Count of people with Parents or Children Onboard')



Most people had no Children or Parent/s with them onboard.

[38]:
```
# Now to study survival rate of people with Parent/children onboard with them.
sns.barplot(data=data, x='Parch', y='Survived')
plt.title("Survival rate of passengers based on Parent/Children Feature")
```

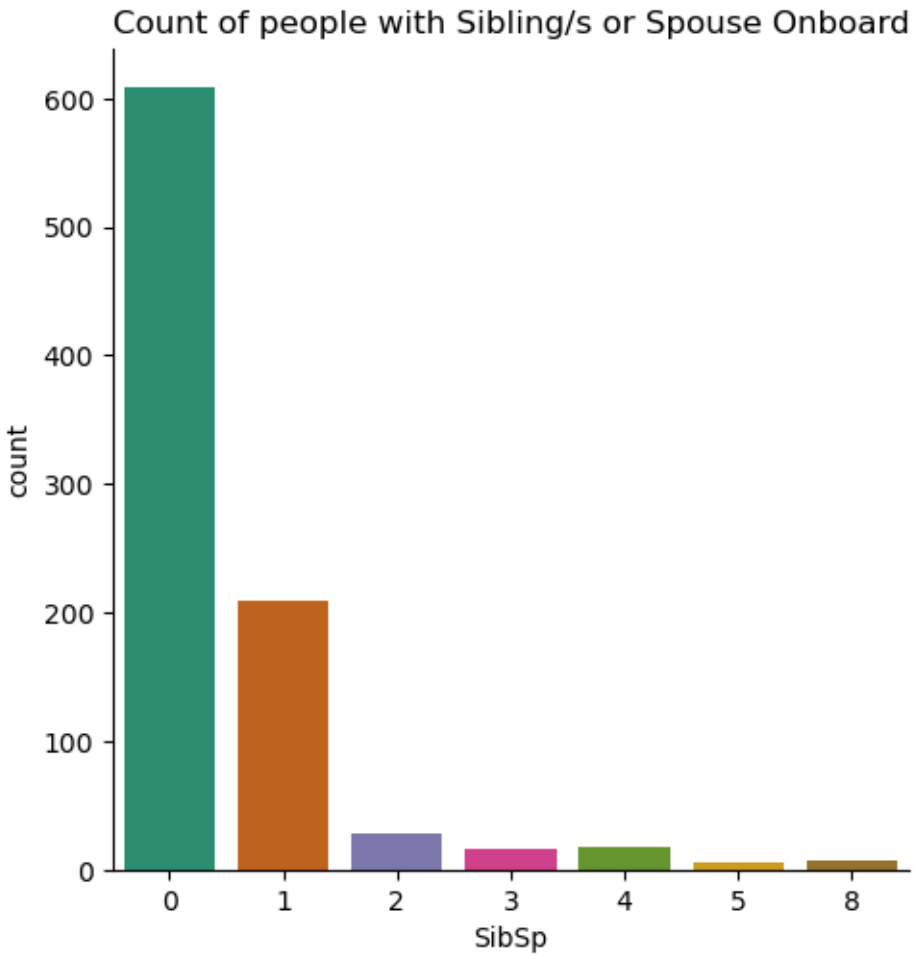
[38]:
```
Text(0.5, 1.0, 'Survival rate of passengers based on Parent/Children Feature')
```



General trend is, if a passenger has parents or children with him, his survival rate chances increased. But the last column (With Prach value=5) doesn't follow the that trend.

[39]:
```
# SibSp column investigation in depth:
sns.catplot(x="SibSp", kind="count", palette="Dark2", data=data)
plt.title("Count of people with Sibling/s or Spouse Onboard")
```

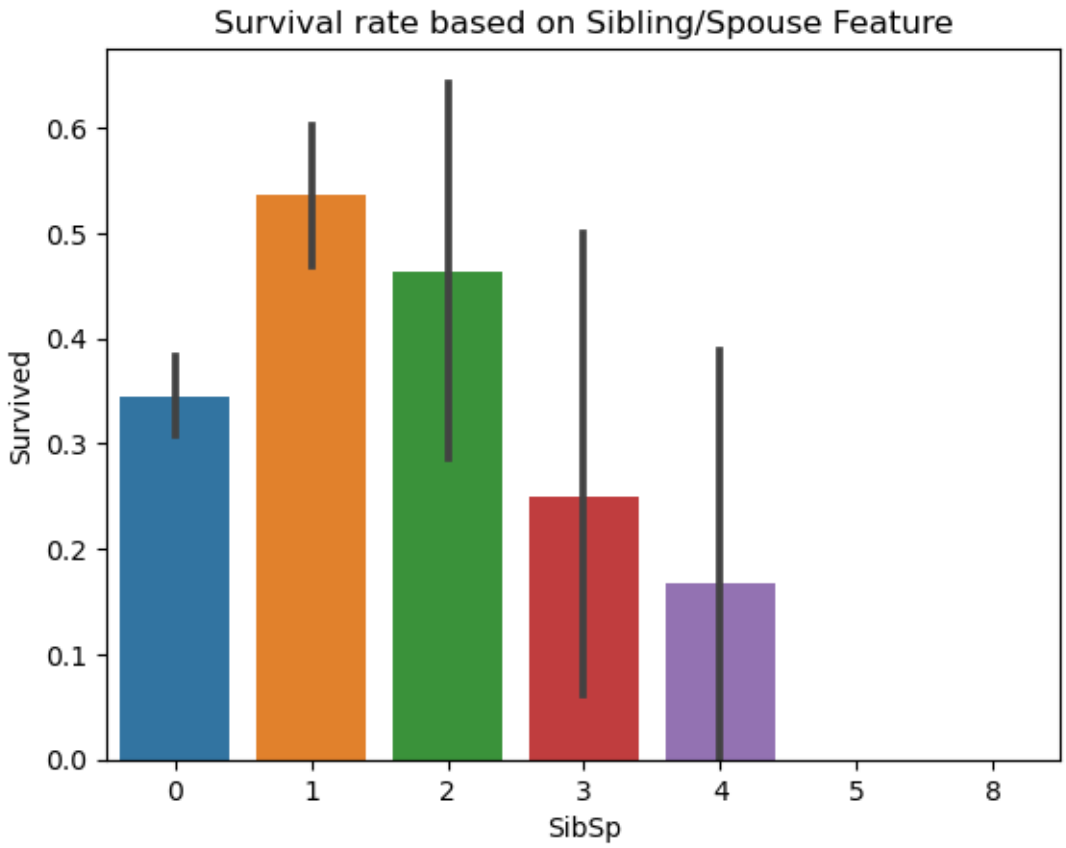
[39]:
```
Text(0.5, 1.0, 'Count of people with Sibling/s or Spouse Onboard')
```



Most people had no Siblings or Spouse with them onboard.

[40]:
```
# Now to study survival rate of people with Siblings/Spouse onboard with them.

sns.barplot(data=data, x='SibSp', y='Survived')
plt.title("Survival rate based on Sibling/Spouse Feature")
```

[40]:
```
Text(0.5, 1.0, 'Survival rate based on Sibling/Spouse Feature')
```



More the number of Siblings/Spouse, lesser the chances of survival.

****Answers to a few questions:****

1. What is the most important factor in determining survival of the Titanic incident?

Ans: Sex is the most important feature in determining survival of the Titanic incident. Female survived way more than male. That means female were given priority on life boats even though there were more male passenegers onboard than female.

2. In the movie, the upper-class passengers were given preference on lifeboats. Does this show in the data?

Ans: Yes, as can be seen from EDA done above passenegers who paid highest Fare and were in upper class 1, survived more than the passengers who paid less fare and were in class 2 or 3.

3. "Women and children first". Was this the case?

Ans: Yes, female and younger children were given preference on the life boats which i have established after detailed EDA of Sex and Age columns. Passengers in the age bracket 0-10 years survived the most.

4. Add one other observation that you have noted in the dataset.

Ans: Passengers who embraked from Cherbourg survived the most and passengers who embarked

from Southampton town survived the least. Since we already know from Fare column analysis that majority of the people bought cheapest tickets so we can establish the fact that people who embarked the ship from Southampton paid the least Fare and therefore survived the least. This i have establised by thorough investigation of Embarked column.