Before you turn this problem in, make sure everything runs as expected. First, **restart the kernel** (in the menu bar, select Kernel$\rightarrow$Restart) and then **run all cells** (in the menu bar, select Cell$\rightarrow$Run All).

Make sure that in addition to the code, you provide written answers for all questions of the assignment.

Below, please fill in your name and collaborators:

In [1]:
```
NAME = "Jordan Vercillo"
COLLABORATORS = "Jordan Vercillo"
```

Assignment 2 - Data Analysis using Pandas

(15 points total)

For this assignment, we will analyze the open dataset with data on the passengers aboard the Titanic.

The data file for this assignment can be downloaded from Kaggle website: https://www.kaggle.com/c/titanic/data, file `train.csv`. It is also attached to the assignment page. The definition of all variables can be found on the same Kaggle page, in the Data Dictionary section.

Read the data from the file into pandas DataFrame. Analyze, clean and transform the data to answer the following question:

What categories of passengers were most likely to survive the Titanic disaster?

Question 1. *(4 points)*

- The answer to the main question - What categories of passengers were most likely to survive the Titanic disaster? *(2 points)*
- The detailed explanation of the logic of the analysis *(2 points)*

Question 2. *(3 points)*

- What other attributes did you use for the analysis? Explain how you used them and why you decided to use them.
- Provide a complete list of all attributes used.

Question 3. *(3 points)*

- Did you engineer any attributes (created new attributes)? If yes, explain the rationale and how the new attributes were used in the analysis?

- If you have excluded any attributes from the analysis, provide an explanation why you believe they can be excluded.

Question 4. *(5 points)*

- How did you treat missing values for those attributes that you included in the analysis (for example, `age` attribute)? Provide a detailed explanation in the comments.

Question1.__My_Analysis

RESULT: The category that had the highest chance of survival (with a number of passengers greater than 10, below 10 is too much of an outlier), is wealthy females with or without kids

I tested a bunch of variables including Age, Gender, Port of Embarkation, Ticket Class, Age, Traveling alone or with Family, Traveling with children and Family size.

I came to a few meaniful realizations like females were most likely to survive (74% survival rate), Children (51% survival rate),1st class (62% survival rate), Traveling with Kids (51% survival rate)

Adult (18-60) Females in first class traveling without kids (98%)

Adult (18-60) Females in first class traveling with kids (95%)

Female Children with kids, young moms, also had a 100% chance of survival but there wasn't too many, about 11

First I analyzed the data, .info(), Describe(), I checked out what it looked like with .head(), experimented with making passenger ID the index but decided I didn't need it. I also renamed the columns to make it easier to read. I then had to impute or remove all the missing values then find the survival rate and test certain values against the survival rate. I made sure to format the data as a data frame (for better readabliity) include a count (To see how many vlaues are there and if it's statisically significant), and sort it by decending based on my key variable "survived"

After I tested all the variables individually I started to group them by common variables like ['AgeGroup', 'TicketClass', 'TravelersWithKids', 'Gender']. I tried looking at it in a few different ways to see if I could find anything meaniful about the analysis. I made sure to pay more attention to results that had enough of a sample size to be accurate, ie. males without kids in 3rd class had 273 data points and a 12% chance of survival while females without kids had 57 data points and a 98% chance of survival. Something like Senior 60+ in 3rd class female with 100% chance of survival I counted as an outlier because I didn't feel that there was a statisically significant size of samples.

That reasoning led to me modify my analysis when needed like when I initally created the age variables to include infants, toddlers, teens, but then realized that there wasn't enough there to make any meaniful inferences. (It did look like most infants survived 85% chance which was nice to see :))

Question2.

0 PassengerId 889 non-null int64

1 Survived 889 non-null int64

2 TicketClass 889 non-null int64

3 Name 889 non-null object

4 Gender 889 non-null object

5 Age 889 non-null float64

6 SiblingsSpousesAboard 889 non-null int64

7 ParentsChildrenAboard 889 non-null int64

8 TicketNumber 889 non-null object

9 PassengerFare 889 non-null float64

10 PortOfEmbarkation 889 non-null object

11 Title 889 non-null object

12 LastName 889 non-null object

13 TravelersWithKids 889 non-null int32

14 Traveling Alone 889 non-null int32

15 FamilySize 889 non-null int64

USED IN ANALYSIS

I used Survived as the main indicator of whether the Passenger Survived

I used ticketClass to see what class there were seated in to see if that impacted survival, I also used this as one of the most important indicators for my final analysis

Name was mainly to extract data for last name and title to be used in cleaning the analysis

I used Gender to see who had a higher chance of survival Male or Female, I also

used this as one of the most important indicators for my final analysis

I used Age to see if children, senior or adults had the highest chance of survival, I also used this as one of the most important indicators for my final analysis

SiblingsSpousesAboard was used to see if the passenger was traveling by themselves along with ParentsChildrenAboard

ParentsChildrenAboard was primarily used to see if the passenger was traveling with Kids or not

TravelersWithKids was used to see if families or parents had a higher chance of survival

NOT USED IN ANALYSIS

Passenger ID was not used in the analysis because we we're looking at specific passengers but grouping them into categories

Ticket Number was used to see family size however this was not used in the final analysis

Passenger fare could be used to indicate how wealthy a person but TicketClass also provided that information and was better organized for analysis

Port of Embarkation I took a look at but didn't show anything that stood out too much.

Question3.

11 Title 889 non-null object

12 LastName 889 non-null object

13 TravelersWithKids 889 non-null int32

14 Traveling Alone 889 non-null int32

15 FamilySize 889 non-null int64

I created a couple of attributes, however I did not use too many of them for the analysis aside from Traveling with Kids.

11 Title (Created to help impute the value of age more accurately, I removed the last name from the name attribute, lambda x: x.split(','), then I was able to isloate the title as each title ended with a "." apply(lambda x: x[1].split('.')[0].strip(). I felt that using

title more accurately imputed the age as title is more tied to your age, ie. DR. older, Master is another name for a younger person.)

12 LastName (To try and find family size but this would introduce potential errors if someone had the same last name but wasn't traveling together)

13 TravelersWithKids (The parents column I tested to see if this column was greater than 0 which would indicate that they were traveling with kids, I didn't need to know the family size just if they were parents with kids or not)

14 Traveling Alone (I used a calc to see if SibSp and Parch == 0 because this would indicate if they are traveling alone or with someone, family, kids, parents)

15 FamilySize (I tried to see if family size had an impact on survivalbility, I used the same ticket number to check as I noticed each family last name had a common ticket number. I found that families 2-4 were the sweet spot for suvivability but any less or more would reduce your survivability.)

Question4.

First I dropped the 2 missing values from PortOfEmbarkation, this wouldn't impact the results of my analysis

Dropping the 2 missing values from PortOfEmbarkation

Titanic.dropna(subset=['PortOfEmbarkation'], inplace=True)

I then created a variable Name_Split to split the "Name" attribute into 2 sections, 1 with the Last name before the "," and another section after. I then noticed that the title ended with a "." so I created my second attribute "Title which would select the 2nd section ie. Braund, / Mr. Owen Haris, then split it again but kept the first second ie. Braund, / Mr. / Owen Haris. Finally I droped the new attribute and kept Title.

Titanic['Name_Split'] = Titanic['Name'].apply(lambda x: x.split(',')) Titanic['Title'] = Titanic['Name_Split'].apply(lambda x: x[1].split('.')[0].strip()) Titanic.drop('Name_Split', axis=1, inplace=True)

I then created a median_ages variable that would provide the medium value of all things I felt were important like my newly created title attribute, Ticket Class and Gender. This would be better than something like the mean as mean could include outliers like 80+ or below 5 which could skew the accurate guess of age we

were looking for. median seemed like a more accurate calcuation

median_ages = Titanic.groupby(['Title', 'TicketClass', 'Gender'])['Age'].median() median_ages

I then had to create an impute_age function which would look for missing values in the "Age" column and return the medium age using the medium_age variable created previously using the pd.isnull function.

Finally I applied this function to the age column and effected the entire row by including axis = 1

Imputing Age based on the medium value of groupby, title, gender, and ticketclass

def impute_age(row): if pd.isnull(row['Age']): return median_ages[row['Title'], row['TicketClass'], row['Gender']] else: return row['Age']

Titanic['Age'] = Titanic.apply(impute_age, axis=1)

MAIN ANALYSIS BELOW

In [2]:
```
import numpy as np
import pandas as pd
```

In [3]:
```
Titanic = pd.read_csv("train.csv")
Titanic
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	F
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2!
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0!
...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4!
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7!

891 rows × 12 columns

In [4]:

```
Titanic.head()
```

Out[4]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.250C
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.925C
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.100C
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.050C

In [5]:
```
Titanic.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 [6]:
```
Titanic.describe()
```

Out[6]:

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.00000(
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.69342S
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.00000(
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.91040(
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.45420(
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.00000(
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.32920(

In [7]:
```
Titanic["Survived"].describe()
```

Out[7]:
```
count    891.000000
mean       0.383838
std        0.486592
min        0.000000
25%        0.000000
50%        0.000000
75%        1.000000
max        1.000000
Name: Survived, dtype: float64
```

In [8]:
```
S_Rate = (Titanic["Survived"].mean())*100
f_s_rate = "{:.2f}%".format(S_Rate)

print("The average survival rate is :", f_s_rate)
```

```
The average survival rate is : 38.38%
```

In [9]:
```
Titanic.rename(columns={
    'survival': 'Survived',
    'Pclass': 'TicketClass',
    'Sex': 'Gender',
    'SibSp': 'SiblingsSpousesAboard',
    'Parch': 'ParentsChildrenAboard',
    'Ticket': 'TicketNumber',
    'Fare': 'PassengerFare',
    'Cabin': 'CabinNumber',
    'Embarked': 'PortOfEmbarkation'
}, inplace=True)
```

In [10]:
```
Titanic.set_index('PassengerId', inplace=True)
```

In [11]:
```
Titanic.head()
```

Out[11]:

	Survived	TicketClass	Name	Gender	Age	SiblingsSpousesAboard	Paren
PassengerId							
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	
5	0	3	Allen, Mr. William Henry	male	35.0	0	

In [12]:
```
Titanic.reset_index(inplace=True)
```

In [13]:
```
Titanic.head()
```

Out[13]:

	PassengerId	Survived	TicketClass	Name	Gender	Age	SiblingsSpousesAboard	Pa
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	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	

In [14]:

```
Titanic.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   TicketClass            891 non-null    int64  
 3   Name                   891 non-null    object 
 4   Gender                 891 non-null    object 
 5   Age                    714 non-null    float64
 6   SiblingsSpousesAboard  891 non-null    int64  
 7   ParentsChildrenAboard  891 non-null    int64  
 8   TicketNumber           891 non-null    object 
 9   PassengerFare          891 non-null    float64
 10  CabinNumber            204 non-null    object 
 11  PortOfEmbarkation      889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

This is cleaning of the dataset and imputing of NaN values

In [15]:
```
#Dropping the 2 missing values from PortOfEmbarkation
Titanic.dropna(subset=['PortOfEmbarkation'], inplace=True)
```

In [16]:
```
#Splitting Name to find the title to get a more accurate imputation of age

Titanic['Name_Split'] = Titanic['Name'].apply(lambda x: x.split(','))
Titanic['Title'] = Titanic['Name_Split'].apply(lambda x: x[1].split('.')[0].strip()
Titanic.drop('Name_Split', axis=1, inplace=True)

print(Titanic['Title'].value_counts())
```

```
Title
Mr              517
Miss            181
Mrs             124
Master           40
Dr                7
Rev               6
Mlle              2
Major             2
Col               2
the Countess      1
Capt              1
Ms                1
Sir               1
Lady              1
Mme               1
Don               1
Jonkheer          1
Name: count, dtype: int64
```

In [17]:
```
Titanic.head()
```

Out[17]:

	PassengerId	Survived	TicketClass	Name	Gender	Age	SiblingsSpousesAboard	Pa
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	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	

In [18]:

```
median_ages = Titanic.groupby(['Title', 'TicketClass', 'Gender'])['Age'].median()
median_ages
```

Out[18]:
```
Title         TicketClass  Gender
Capt          1            male      70.0
Col           1            male      58.0
Don           1            male      40.0
Dr            1            female    49.0
                           male      44.0
              2            male      38.5
Jonkheer      1            male      38.0
Lady          1            female    48.0
Major         1            male      48.5
Master        1            male       4.0
              2            male       1.0
              3            male       4.0
Miss          1            female    30.0
              2            female    24.0
              3            female    18.0
Mlle          1            female    24.0
Mme           1            female    24.0
Mr            1            male      40.0
              2            male      31.0
              3            male      26.0
Mrs           1            female    40.0
              2            female    32.0
              3            female    31.0
Ms            2            female    28.0
Rev           2            male      46.5
Sir           1            male      49.0
the Countess  1            female    33.0
Name: Age, dtype: float64
```

In [19]:
```
#Imputing Age based on the medium value of groupby, title, gender, and ticketclass
def impute_age(row):
    if pd.isnull(row['Age']):
        return median_ages[row['Title'], row['TicketClass'], row['Gender']]
    else:
        return row['Age']

Titanic['Age'] = Titanic.apply(impute_age, axis=1)
```

In [20]:
```
#Dropping CabinNumber as there's too many missing values and TicketClass can provid
Titanic.drop('CabinNumber', axis=1, inplace=True)
```

In [21]:
```
#Creating Last name column
Titanic['LastName'] = Titanic['Name'].apply(lambda x: x.split(',')[0])
```

In [22]:
```
#Traveling with Kids
Titanic['TravelersWithKids'] = (Titanic['ParentsChildrenAboard']>0).astype(bool)
```

In [23]:
```
#Traveling Alone?
Titanic['Traveling Alone'] = ((Titanic['ParentsChildrenAboard']+Titanic['SiblingsSp
```

In [24]:
```
#Family Size based on same ticket number, this could be inaccurate due to sharing t
Titanic['FamilySize'] = Titanic.groupby('TicketNumber')['TicketNumber'].transform('
```

In [25]:
```
Titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 889 entries, 0 to 890
Data columns (total 16 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   PassengerId             889 non-null    int64  
 1   Survived                889 non-null    int64  
 2   TicketClass             889 non-null    int64  
 3   Name                    889 non-null    object 
 4   Gender                  889 non-null    object 
 5   Age                     889 non-null    float64
 6   SiblingsSpousesAboard   889 non-null    int64  
 7   ParentsChildrenAboard   889 non-null    int64  
 8   TicketNumber            889 non-null    object 
 9   PassengerFare           889 non-null    float64
 10  PortOfEmbarkation       889 non-null    object 
 11  Title                   889 non-null    object 
 12  LastName                889 non-null    object 
 13  TravelersWithKids       889 non-null    bool   
 14  Traveling Alone         889 non-null    int32  
 15  FamilySize              889 non-null    int64  
dtypes: bool(1), float64(2), int32(1), int64(6), object(6)
memory usage: 108.5+ KB
```

In [26]:
```
Titanic
```

Out[26]:

	PassengerId	Survived	TicketClass	Name	Gender	Age	SiblingsSpousesAboard
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	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1
4	5	0	3	Allen, Mr. William Henry	male	35.0	0
...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	18.0	1
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0

889 rows × 16 columns

Creating groupings of categories

GenderGroup: (Male/Female)

Port of Embarkation: (C/Q/S)

AgeGroup: ('Child' 0-12, 'Teen'12-18, 'Adult' 18-60, 'Senior' 60+)

TicketClassGroup: Wealth Status (1,2,3)

Traveling Alone: Single or Family (is parentsaboard and siblingsaboard is equal to 0)

ParentswithChildren: Parents column = 1

FamilySizegroup : value counts based on Last Name and Ticket Number

In [27]:
```
print("The average survival rate is :", f_s_rate)
```

```
The average survival rate is : 38.38%
```

In [28]:
```
#Gender Group
gender_survival = Titanic.groupby('Gender')['Survived'].mean().sort_values(ascendin
gender_survival
```

Out[28]:
```
Gender
female    0.740385
male      0.188908
Name: Survived, dtype: float64
```

In [29]:
```
print("Females are most likely to survive")
```

```
Females are most likely to survive
```

In [30]:
```
port_survival = Titanic.groupby('PortOfEmbarkation')['Survived'].mean().sort_values
```

```
port_survival
```

Out[30]:
```
PortOfEmbarkation
C    0.553571
Q    0.389610
S    0.336957
Name: Survived, dtype: float64
```

In [31]:
```
print("Port C Most likely to survive")
```

```
Port C Most likely to survive
```

In [32]:
```
bins = [0,1,3, 12, 18, 60, 120]
labels = ['Infant','Toddlers','Child', 'Teen', 'Adult', 'Senior']
Titanic['AgeGroup'] = pd.cut(Titanic['Age'], bins=bins, labels=labels)
Titanic.groupby('AgeGroup')['Survived'].mean().sort_values(ascending=False)
```

```
C:\Users\jverc\AppData\Local\Temp\ipykernel_7632\2220056522.py:4: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  Titanic.groupby('AgeGroup')['Survived'].mean().sort_values(ascending=False)
```

Out[32]:
```
AgeGroup
Infant      0.857143
Child       0.511628
Toddlers    0.500000
Teen        0.475728
Adult       0.354046
Senior      0.190476
Name: Survived, dtype: float64
```

In [33]:
```
bins = [0, 18, 60, 120]
labels = ['Children 0-18','Adult 12-60', 'Senior 60+']
Titanic['AgeGroup'] = pd.cut(Titanic['Age'], bins=bins, labels=labels)
Titanic.groupby('AgeGroup')['Survived'].mean().sort_values(ascending=False)
```

```
C:\Users\jverc\AppData\Local\Temp\ipykernel_7632\2072681962.py:4: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  Titanic.groupby('AgeGroup')['Survived'].mean().sort_values(ascending=False)
```

Out[33]:
```
AgeGroup
Children 0-18    0.517045
Adult 12-60      0.354046
Senior 60+       0.190476
Name: Survived, dtype: float64
```

In [34]:
```
age_group_survival = Titanic.groupby('AgeGroup')['Survived'].mean()
sorted_survival_rates = age_group_survival.sort_values(ascending=False)
age_group_counts = Titanic.groupby('AgeGroup')['Survived'].count()
age_group_stats = pd.DataFrame({
    'Survival Rate': sorted_survival_rates,
    'Count': age_group_counts
}).sort_values(by='Survival Rate', ascending=False)
age_group_stats
```

```
C:\Users\jverc\AppData\Local\Temp\ipykernel_7632\2796588955.py:1: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  age_group_survival = Titanic.groupby('AgeGroup')['Survived'].mean()
C:\Users\jverc\AppData\Local\Temp\ipykernel_7632\2796588955.py:3: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  age_group_counts = Titanic.groupby('AgeGroup')['Survived'].count()
```

Out[34]:

AgeGroup	Survival Rate	Count
Children 0-18	0.517045	176
Adult 12-60	0.354046	692
Senior 60+	0.190476	21

In [35]:
```
print("under 18 are most likely to survive")
```

```
under 18 are most likely to survive
```

In [36]:
```
ticket_class_survival = Titanic.groupby('TicketClass')['Survived'].mean().sort_valu
ticket_class_survival
```

Out[36]:
```
TicketClass
1    0.626168
2    0.472826
3    0.242363
Name: Survived, dtype: float64
```

In [37]:
```
print("1st Class are most likely to survive")
```

```
1st Class are most likely to survive
```

In [38]:
```
Titanic.groupby('Traveling Alone')['Survived'].mean().sort_values(ascending=False)
```

Out[38]:
```
Traveling Alone
0    0.505650
1    0.300935
Name: Survived, dtype: float64
```

In [39]:
```
print("Traveling with Family most likely to survive")
```

```
Traveling with Family most likely to survive
```

In [40]:
```
parents_survival = Titanic.groupby('TravelersWithKids')['Survived'].mean().sort_val
parents_survival
```

Out[40]:
```
TravelersWithKids
True     0.511737
False    0.341716
Name: Survived, dtype: float64
```

In [41]:
```
print("Traveling with Children most likely to survive")
```

```
Traveling with Children most likely to survive
```

In [42]:
```
Titanic.groupby('FamilySize')['Survived'].mean().sort_values(ascending=False)
```

Out[42]:
```
FamilySize
3    0.698413
2    0.569892
4    0.500000
1    0.297989
7    0.238095
5    0.000000
6    0.000000
Name: Survived, dtype: float64
```

In [43]:
```
print("Family size of 2-4 are most likely to survive, while traveling alone or with
```

```
Family size of 2-4 are most likely to survive, while traveling alone or with too big
of a family reduced your survivability
```

In [44]:
```
Titanic.groupby('Title')['Survived'].mean().sort_values(ascending=False)
```

Out[44]:
```
Title
the Countess    1.000000
Mlle            1.000000
Sir             1.000000
Ms              1.000000
Lady            1.000000
Mme             1.000000
Mrs             0.790323
Miss            0.696133
Master          0.575000
Col             0.500000
Major           0.500000
Dr              0.428571
Mr              0.156673
Jonkheer        0.000000
Rev             0.000000
Don             0.000000
Capt            0.000000
Name: Survived, dtype: float64
```

In [45]:
```
survival_rate = Titanic.groupby('Title')['Survived'].mean()
title_counts = Titanic.groupby('Title')['Survived'].count()
survival_rate_df = survival_rate.reset_index(name='Survival Rate')
title_counts_df = title_counts.reset_index(name='Count')
title_stats = pd.merge(survival_rate_df, title_counts_df, on='Title')
title_stats_sorted = title_stats.sort_values(by='Survival Rate', ascending=False)
title_stats_sorted
```

Out[45]:

	Title	Survival Rate	Count
16	the Countess	1.000000	1
9	Mlle	1.000000	2
15	Sir	1.000000	1
13	Ms	1.000000	1
5	Lady	1.000000	1
10	Mme	1.000000	1
12	Mrs	0.790323	124
8	Miss	0.696133	181
7	Master	0.575000	40
1	Col	0.500000	2
6	Major	0.500000	2
3	Dr	0.428571	7
11	Mr	0.156673	517
4	Jonkheer	0.000000	1
14	Rev	0.000000	6
2	Don	0.000000	1
0	Capt	0.000000	1

In [46]:
```
print("Mrs, Miss, and Master most likely to survive while Mr had a 15% survival rat
```

```
Mrs, Miss, and Master most likely to survive while Mr had a 15% survival rate
```

In [47]:
```
multi_grouped = Titanic.groupby(['AgeGroup', 'TicketClass','TravelersWithKids','Gen
multi_grouped
```

```
C:\Users\jverc\AppData\Local\Temp\ipykernel_7632\258836269.py:1: FutureWarning: The
default of observed=False is deprecated and will be changed to True in a future vers
ion of pandas. Pass observed=False to retain current behavior or observed=True to ad
opt the future default and silence this warning.
  multi_grouped = Titanic.groupby(['AgeGroup', 'TicketClass','TravelersWithKids','Ge
nder'])['Survived'].mean().unstack().sort_index(ascending=False)
```

Out[47]:

		Gender	female	male
AgeGroup	**TicketClass**	**TravelersWithKids**		
Senior 60+	**3**	**True**	NaN	NaN
		False	1.000000	0.000000
	2	**True**	NaN	NaN
		False	NaN	0.333333
	1	**True**	NaN	0.000000
		False	1.000000	0.111111
Adult 12-60	**3**	**True**	0.375000	0.066667
		False	0.488372	0.120879
	2	**True**	0.950000	0.000000
		False	0.880952	0.087500
	1	**True**	0.956522	0.312500
		False	0.982456	0.393258
Children 0-18	**3**	**True**	0.371429	0.277778
		False	0.682927	0.157895
	2	**True**	1.000000	1.000000
		False	1.000000	0.000000
	1	**True**	0.857143	1.000000
		False	1.000000	0.000000

In [48]:

```
grouped = Titanic.groupby(['AgeGroup', 'TicketClass', 'TravelersWithKids', 'Gender'
    SurvivalRate=('Survived', 'mean'),
    Count=('Survived', 'count')
).reset_index()


filtered_groups = grouped[grouped['Count'] > 0]
sorted_groups = filtered_groups.sort_values(by='SurvivalRate', ascending=False)
sorted_groups
```

```
C:\Users\jverc\AppData\Local\Temp\ipykernel_7632\1131773453.py:1: FutureWarning: The
default of observed=False is deprecated and will be changed to True in a future vers
ion of pandas. Pass observed=False to retain current behavior or observed=True to ad
opt the future default and silence this warning.
  grouped = Titanic.groupby(['AgeGroup', 'TicketClass', 'TravelersWithKids', 'Gende
r']).agg(
```

Out[48]:

	AgeGroup	TicketClass	TravelersWithKids	Gender	SurvivalRate	Count
0	Children 0-18	1	False	female	1.000000	4
3	Children 0-18	1	True	male	1.000000	4
4	Children 0-18	2	False	female	1.000000	3
32	Senior 60+	3	False	female	1.000000	1
6	Children 0-18	2	True	female	1.000000	11
7	Children 0-18	2	True	male	1.000000	9
24	Senior 60+	1	False	female	1.000000	1
12	Adult 12-60	1	False	female	0.982456	57
14	Adult 12-60	1	True	female	0.956522	23
18	Adult 12-60	2	True	female	0.950000	20
16	Adult 12-60	2	False	female	0.880952	42
2	Children 0-18	1	True	female	0.857143	7
8	Children 0-18	3	False	female	0.682927	41
20	Adult 12-60	3	False	female	0.488372	43
13	Adult 12-60	1	False	male	0.393258	89
22	Adult 12-60	3	True	female	0.375000	24
10	Children 0-18	3	True	female	0.371429	35
29	Senior 60+	2	False	male	0.333333	3
15	Adult 12-60	1	True	male	0.312500	16
11	Children 0-18	3	True	male	0.277778	36
9	Children 0-18	3	False	male	0.157895	19
21	Adult 12-60	3	False	male	0.120879	273
25	Senior 60+	1	False	male	0.111111	9
17	Adult 12-60	2	False	male	0.087500	80
23	Adult 12-60	3	True	male	0.066667	15
19	Adult 12-60	2	True	male	0.000000	10
1	Children 0-18	1	False	male	0.000000	1
27	Senior 60+	1	True	male	0.000000	3
5	Children 0-18	2	False	male	0.000000	6
33	Senior 60+	3	False	male	0.000000	4

In [49]:
```
grouped = Titanic.groupby(['AgeGroup', 'TicketClass', 'TravelersWithKids', 'Gender'
    SurvivalRate=('Survived', 'mean'),
    Count=('Survived', 'count')
).reset_index()


filtered_groups = grouped[grouped['Count'] > 10]
sorted_groups = filtered_groups.sort_values(by='SurvivalRate', ascending=False)
sorted_groups
```

```
C:\Users\jverc\AppData\Local\Temp\ipykernel_7632\2320419055.py:1: FutureWarning: The
default of observed=False is deprecated and will be changed to True in a future vers
ion of pandas. Pass observed=False to retain current behavior or observed=True to ad
opt the future default and silence this warning.
  grouped = Titanic.groupby(['AgeGroup', 'TicketClass', 'TravelersWithKids', 'Gende
r']).agg(
```

Out[49]:

	AgeGroup	TicketClass	TravelersWithKids	Gender	SurvivalRate	Count
6	Children 0-18	2	True	female	1.000000	11
12	Adult 12-60	1	False	female	0.982456	57
14	Adult 12-60	1	True	female	0.956522	23
18	Adult 12-60	2	True	female	0.950000	20
16	Adult 12-60	2	False	female	0.880952	42
8	Children 0-18	3	False	female	0.682927	41
20	Adult 12-60	3	False	female	0.488372	43
13	Adult 12-60	1	False	male	0.393258	89
22	Adult 12-60	3	True	female	0.375000	24
10	Children 0-18	3	True	female	0.371429	35
15	Adult 12-60	1	True	male	0.312500	16
11	Children 0-18	3	True	male	0.277778	36
9	Children 0-18	3	False	male	0.157895	19
21	Adult 12-60	3	False	male	0.120879	273
17	Adult 12-60	2	False	male	0.087500	80
23	Adult 12-60	3	True	male	0.066667	15

In [50]:
```
grouped = Titanic.groupby(['AgeGroup','TravelersWithKids', 'Gender']).agg(
    SurvivalRate=('Survived', 'mean'),
    Count=('Survived', 'count')
).reset_index()


filtered_groups = grouped[grouped['Count'] > 0]
```

```
sorted_groups = filtered_groups.sort_values(by='SurvivalRate', ascending=False)
sorted_groups
```

```
C:\Users\jverc\AppData\Local\Temp\ipykernel_7632\2606926184.py:1: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  grouped = Titanic.groupby(['AgeGroup','TravelersWithKids', 'Gender']).agg(
```

Out[50]:

	AgeGroup	TravelersWithKids	Gender	SurvivalRate	Count
8	Senior 60+	False	female	1.000000	2
4	Adult 12-60	False	female	0.802817	142
6	Adult 12-60	True	female	0.746269	67
0	Children 0-18	False	female	0.729167	48
2	Children 0-18	True	female	0.566038	53
3	Children 0-18	True	male	0.469388	49
5	Adult 12-60	False	male	0.169683	442
7	Adult 12-60	True	male	0.146341	41
9	Senior 60+	False	male	0.125000	16
1	Children 0-18	False	male	0.115385	26
11	Senior 60+	True	male	0.000000	3

In [51]:

```
grouped = Titanic.groupby(['AgeGroup','TicketClass', 'Gender']).agg(
    SurvivalRate=('Survived', 'mean'),
    Count=('Survived', 'count')
).reset_index()


filtered_groups = grouped[grouped['Count'] > 0]
sorted_groups = filtered_groups.sort_values(by='SurvivalRate', ascending=False)
sorted_groups
```

```
C:\Users\jverc\AppData\Local\Temp\ipykernel_7632\1257758034.py:1: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  grouped = Titanic.groupby(['AgeGroup','TicketClass', 'Gender']).agg(
```

Out[51]:

	AgeGroup	TicketClass	Gender	SurvivalRate	Count
2	Children 0-18	2	female	1.000000	14
16	Senior 60+	3	female	1.000000	1
12	Senior 60+	1	female	1.000000	1
6	Adult 12-60	1	female	0.975000	80
0	Children 0-18	1	female	0.909091	11
8	Adult 12-60	2	female	0.903226	62
1	Children 0-18	1	male	0.800000	5
3	Children 0-18	2	male	0.600000	15
4	Children 0-18	3	female	0.539474	76
10	Adult 12-60	3	female	0.447761	67
7	Adult 12-60	1	male	0.380952	105
15	Senior 60+	2	male	0.333333	3
5	Children 0-18	3	male	0.236364	55
11	Adult 12-60	3	male	0.118056	288
13	Senior 60+	1	male	0.083333	12
9	Adult 12-60	2	male	0.077778	90
17	Senior 60+	3	male	0.000000	4

In [ ]: