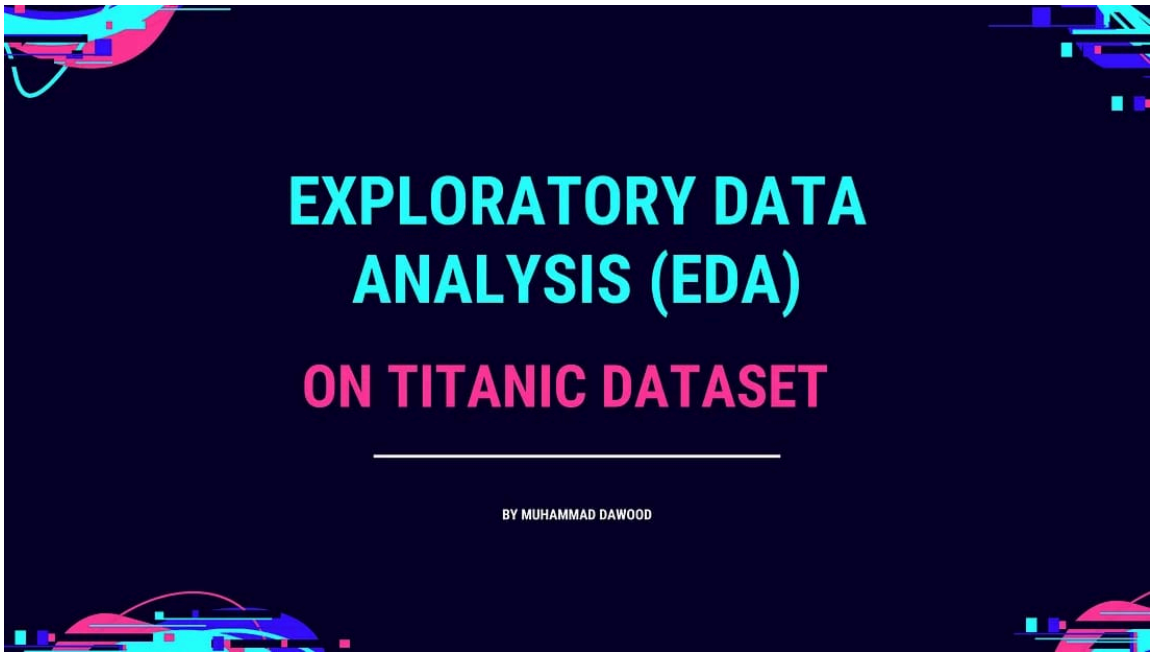
ANALYST: SORNAPUDI PUJITHA

EMAIL:sornapudipujitha@gmail.com

LINKEDIN:PUJITHA SORNAPUDI



INTRODUCTION:

The Titanic dataset is popular for data analysis. It contains information about the passengers onboard the Titanic, including features like age, gender, fare, cabin, and survival status. We will perform exploratory data analysis (EDA) on the Titanic dataset using Python in this project.

DATSET:

The Titanic dataset is available in Seaborn as the 'titanic' dataset. It consists of the following columns:

- **Survived:** Survival status (0 = No, 1 = Yes)
- **Pclass:** Passenger class (1 = 1st class, 2 = 2nd class, 3 = 3rd class)
- **Sex:** Passenger's gender
- **Age:** Passenger's age
- **SibSp:** Number of siblings/spouses aboard
- **Parch:**Number of parents/children aboard
- **Fare:** Fare paid for the ticket
- **Embarked:** Port of embarkation (C = Cherbourg, Q = Queenstown, S = Southampton)
- **Class:** Equivalent to Pclass (1 = 1st class, 2 = 2nd class, 3 = 3rd class)
- **Who:** Passenger's category (man, woman, child)

- **Adult_male:** Whether the passenger is an adult male or not (True or False)
- **Deck:** Cabin deck
- **Embark_town:** Port of embarkation (Cherbourg, Queenstown, Southampton)
- **Alone:** Whether the passenger is alone or not (True or False)
- **Alive:** Survival status (yes or no)

IMPORT LIBRARIES

In [ ]:
```
import seaborn as sns
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

LOADING DATASET

In [ ]:
```
df=sns.load_dataset('titanic')
```

In [ ]:
```
# df.to_csv('titanic.csv',index=False)
# print('Titanic dataset has been saved as titanic.csv')
```

In [ ]:
```
df
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adu
0	0	3	male	22.0	1	0	7.2500	S	Third	man	
1	1	1	female	38.0	1	0	71.2833	C	First	woman	
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	
3	1	1	female	35.0	1	0	53.1000	S	First	woman	
4	0	3	male	35.0	0	0	8.0500	S	Third	man	
...	...	...	...	...	...	...	...	...	...	...	
886	0	2	male	27.0	0	0	13.0000	S	Second	man	
887	1	1	female	19.0	0	0	30.0000	S	First	woman	
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	
889	1	1	male	26.0	0	0	30.0000	C	First	man	
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	

891 rows × 15 columns

EXPLORING DATASET:

In [ ]:
```
df.shape
```

Out[ ]: (891, 15)

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

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_m
0	0	3	male	22.0	1	0	7.2500	S	Third	man	T
1	1	1	female	38.0	1	0	71.2833	C	First	woman	Fa
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	Fa
3	1	1	female	35.0	1	0	53.1000	S	First	woman	Fa
4	0	3	male	35.0	0	0	8.0500	S	Third	man	T

In [ ]:
```
df.tail()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult
886	0	2	male	27.0	0	0	13.00	S	Second	man	
887	1	1	female	19.0	0	0	30.00	S	First	woman	
888	0	3	female	NaN	1	2	23.45	S	Third	woman	
889	1	1	male	26.0	0	0	30.00	C	First	man	
890	0	3	male	32.0	0	0	7.75	Q	Third	man	

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 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
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

In [ ]:
```
df.describe(include='number')
```

Out[ ]:

	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

In [ ]:
```
df.describe(include='object')
```

Out[ ]:

	sex	embarked	who	embark_town	alive
count	891	889	891	889	891
unique	2	3	3	3	2
top	male	S	man	Southampton	no
freq	577	644	537	644	549

In [ ]:
```
df['who'].unique()
```

Out[ ]:
```
array(['man', 'woman', 'child'], dtype=object)
```

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

Out[ ]:

	0
survived	0
pclass	0
sex	0
age	177
sibsp	0
parch	0
fare	0
embarked	2
class	0
who	0
adult_male	0
deck	688
embark_town	2
alive	0
alone	0

dtype: int64

DATA CLEANING

In [ ]:
```
df['age'].isna().head()
```

Out[ ]:

	age
0	False
1	False
2	False
3	False
4	False

dtype: bool

In [ ]:
```
df['age'].isna().tail()
```

Out[ ]:

	age
886	False
887	False
888	True
889	False
890	False

dtype: bool

In [ ]:
```
df['age'].head()
```

Out[ ]:

	age
0	22.0
1	38.0
2	26.0
3	35.0
4	35.0

dtype: float64

In [ ]:
```
avg_age=df['age'].mean()
df['age'].fillna(df['age'].mean(),inplace=True)
```

```
<ipython-input-177-1e1ff30139f4>:2: FutureWarning: A value is trying to be set on a
copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df['age'].fillna(df['age'].mean(),inplace=True)
```

In [ ]:
```
df['age'].isna().sum()
```

Out[ ]:
```
np.int64(0)
```

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

Out[ ]:

	0
survived	0
pclass	0
sex	0
age	0
sibsp	0
parch	0
fare	0
embarked	2
class	0
who	0
adult_male	0
deck	688
embark_town	2
alive	0
alone	0

dtype: int64

In [ ]:
```
df['embarked'].unique()
```

Out[ ]:
```
array(['S', 'C', 'Q', nan], dtype=object)
```

In [ ]:
```
df['embark_town'].unique()
```

Out[ ]:
```
array(['Southampton', 'Cherbourg', 'Queenstown', nan], dtype=object)
```

In [ ]:
```
#check index value of null value in dataframe
index=df[df['embarked'].isnull()]
print(index)
```

```
     survived  pclass     sex   age  sibsp  parch  fare embarked  class  \
61          1       1  female  38.0      0      0  80.0      NaN  First
829         1       1  female  62.0      0      0  80.0      NaN  First

       who  adult_male deck embark_town alive  alone
61   woman       False    B         NaN   yes   True
829  woman       False    B         NaN   yes   True
```

In [ ]:
```
embarked=df['embarked'].mode()
df['embarked'].fillna(df['embarked'].mode()[0],inplace=True)
```

```
<ipython-input-183-2dbe7e15cfa0>:2: FutureWarning: A value is trying to be set on a
copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df['embarked'].fillna(df['embarked'].mode()[0],inplace=True)
```

In [ ]:
```
df.loc[61]
```

Out[ ]:

	61
survived	1
pclass	1
sex	female
age	38.0
sibsp	0
parch	0
fare	80.0
embarked	S
class	First
who	woman
adult_male	False
deck	B
embark_town	NaN
alive	yes
alone	True

dtype: object

In [ ]:
```
df['embarked'].isna().sum()
```

Out[ ]:
```
np.int64(0)
```

In [ ]:
```
df['deck']
```

Out[ ]:

	deck
0	NaN
1	C
2	NaN
3	C
4	NaN
...	...
886	NaN
887	B
888	NaN
889	C
890	NaN

891 rows × 1 columns

dtype: category

In [ ]:

```
index_deck=df[df['deck'].isnull()]
print(index_deck)
```

```
     survived  pclass     sex        age  sibsp  parch     fare embarked  \
0           0       3    male  22.000000      1      0   7.2500        S
2           1       3  female  26.000000      0      0   7.9250        S
4           0       3    male  35.000000      0      0   8.0500        S
5           0       3    male  29.699118      0      0   8.4583        Q
7           0       3    male   2.000000      3      1  21.0750        S
..        ...     ...     ...        ...    ...    ...      ...      ...
884         0       3    male  25.000000      0      0   7.0500        S
885         0       3  female  39.000000      0      5  29.1250        Q
886         0       2    male  27.000000      0      0  13.0000        S
888         0       3  female  29.699118      1      2  23.4500        S
890         0       3    male  32.000000      0      0   7.7500        Q

      class    who  adult_male deck  embark_town alive  alone
0     Third    man        True  NaN  Southampton    no  False
2     Third  woman       False  NaN  Southampton   yes   True
4     Third    man        True  NaN  Southampton    no   True
5     Third    man        True  NaN   Queenstown    no   True
7     Third  child       False  NaN  Southampton    no  False
..      ...    ...         ...  ...          ...   ...    ...
884   Third    man        True  NaN  Southampton    no   True
885   Third  woman       False  NaN   Queenstown    no  False
886  Second    man        True  NaN  Southampton    no   True
888   Third  woman       False  NaN  Southampton    no  False
890   Third    man        True  NaN   Queenstown    no   True

[688 rows x 15 columns]
```

In [ ]:
```
index_deck
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who
0	0	3	male	22.000000	1	0	7.2500	S	Third	mar
2	1	3	female	26.000000	0	0	7.9250	S	Third	womar
4	0	3	male	35.000000	0	0	8.0500	S	Third	mar
5	0	3	male	29.699118	0	0	8.4583	Q	Third	mar
7	0	3	male	2.000000	3	1	21.0750	S	Third	chilc
...	...	...	...	...	...	...	...	...	...	..
884	0	3	male	25.000000	0	0	7.0500	S	Third	mar
885	0	3	female	39.000000	0	5	29.1250	Q	Third	womar
886	0	2	male	27.000000	0	0	13.0000	S	Second	mar
888	0	3	female	29.699118	1	2	23.4500	S	Third	womar
890	0	3	male	32.000000	0	0	7.7500	Q	Third	mar

688 rows × 15 columns

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

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_m
0	0	3	male	22.0	1	0	7.2500	S	Third	man	T
1	1	1	female	38.0	1	0	71.2833	C	First	woman	Fa
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	Fa
3	1	1	female	35.0	1	0	53.1000	S	First	woman	Fa
4	0	3	male	35.0	0	0	8.0500	S	Third	man	T

In [ ]:
```
df['deck'].unique()
```

Out[ ]:
```
[NaN, 'C', 'E', 'G', 'D', 'A', 'B', 'F']
Categories (7, object): ['A', 'B', 'C', 'D', 'E', 'F', 'G']
```

In [ ]:
```
df['deck'].mode()
```

Out[ ]:

	deck
0	C

dtype: category

In [ ]:
```
mode_deck=df['deck'].mode()
df['deck'].fillna(df['deck'].mode()[0],inplace=True)
```

In [ ]:
```
df['deck'].isna().sum()
```

Out[ ]:
```
np.int64(0)
```

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

Out[ ]:

	0
survived	0
pclass	0
sex	0
age	0
sibsp	0
parch	0
fare	0
embarked	0
class	0
who	0
adult_male	0
deck	0
embark_town	2
alive	0
alone	0

dtype: int64

In [ ]:
```
df['embark_town'].value_counts()
```

Out[ ]:

embark_town	count
Southampton	644
Cherbourg	168
Queenstown	77

dtype: int64

In [ ]:
```
df['embark_town'].mode()
```

Out[ ]:

	embark_town
0	Southampton

dtype: object

In [ ]:
```
mode_town=df['embark_town'].mode()
df['embark_town'].fillna(df['embark_town'].mode()[0],inplace=True)
```

```
<ipython-input-197-f7736df52a57>:2: FutureWarning: A value is trying to be set on a
copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df['embark_town'].fillna(df['embark_town'].mode()[0],inplace=True)
```

In [ ]:
```
df
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	whc
0	0	3	male	22.000000	1	0	7.2500	S	Third	mar
1	1	1	female	38.000000	1	0	71.2833	C	First	womar
2	1	3	female	26.000000	0	0	7.9250	S	Third	womar
3	1	1	female	35.000000	1	0	53.1000	S	First	womar
4	0	3	male	35.000000	0	0	8.0500	S	Third	mar
...	...	...	...	...	...	...	...	...	...	..
886	0	2	male	27.000000	0	0	13.0000	S	Second	mar
887	1	1	female	19.000000	0	0	30.0000	S	First	womar
888	0	3	female	29.699118	1	2	23.4500	S	Third	womar
889	1	1	male	26.000000	0	0	30.0000	C	First	mar
890	0	3	male	32.000000	0	0	7.7500	Q	Third	mar

891 rows × 15 columns

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

Out[ ]:

	0
survived	0
pclass	0
sex	0
age	0
sibsp	0
parch	0
fare	0
embarked	0
class	0
who	0
adult_male	0
deck	0
embark_town	0
alive	0
alone	0

dtype: int64

Data Profiling

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

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_m
0	0	3	male	22.0	1	0	7.2500	S	Third	man	T
1	1	1	female	38.0	1	0	71.2833	C	First	woman	Fa
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	Fa
3	1	1	female	35.0	1	0	53.1000	S	First	woman	Fa
4	0	3	male	35.0	0	0	8.0500	S	Third	man	T

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 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
 8   class        891 non-null    category
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         891 non-null    category
 12  embark_town  891 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), category(2), float64(2), int64(4), object(5)
memory usage: 80.7+ KB
```

In [ ]:
```
df.describe(include='number')
```

Out[ ]:

	survived	pclass	age	sibsp	parch	fare
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000
mean	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	0.486592	0.836071	13.002015	1.102743	0.806057	49.693429
min	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	0.000000	2.000000	22.000000	0.000000	0.000000	7.910400
50%	0.000000	3.000000	29.699118	0.000000	0.000000	14.454200
75%	1.000000	3.000000	35.000000	1.000000	0.000000	31.000000
max	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

In [ ]:
```
df.describe(include='object')
```

Out[ ]:

	sex	embarked	who	embark_town	alive
count	891	891	891	891	891
unique	2	3	3	3	2
top	male	S	man	Southampton	no
freq	577	646	537	646	549

In [ ]:
```
df.shape
```

Out[ ]: (891, 15)

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

Out[ ]:

	0
survived	0
pclass	0
sex	0
age	0
sibsp	0
parch	0
fare	0
embarked	0
class	0
who	0
adult_male	0
deck	0
embark_town	0
alive	0
alone	0

dtype: int64

EXPLORATORY DATA ANALYSIS ON DATASET

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

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_m
0	0	3	male	22.0	1	0	7.2500	S	Third	man	T
1	1	1	female	38.0	1	0	71.2833	C	First	woman	Fa
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	Fa
3	1	1	female	35.0	1	0	53.1000	S	First	woman	Fa
4	0	3	male	35.0	0	0	8.0500	S	Third	man	T

In [ ]:

```
# 1. How many passengers were on board the Titanic in this dataset?
df['pclass'].count()
```

Out[ ]:

```
np.int64(891)
```

Insight: Total Passengers travelling in Titanic is 891

In [ ]:

```
# 2. What is the overall surival rate of passengers?
survival_rate=df[df['alive']=='yes']['alive'].value_counts()
survival_rate
Total_Passenger=df['alive'].count()
survival_Percentage=round(survival_rate/Total_Passenger*100,2)
survival_Percentage
```

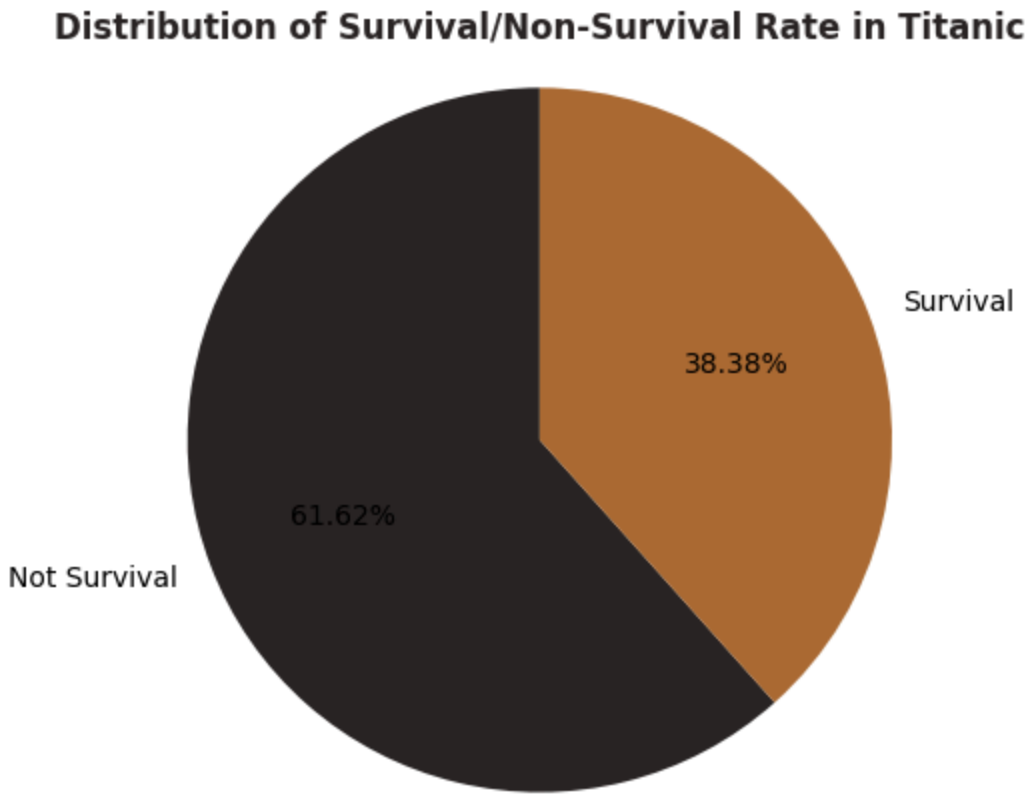
Out[ ]:

alive	count
yes	38.38

dtype: float64

In [ ]:

```
diff=df.groupby('alive')['alive'].value_counts()
diff
Label=['Not Survival','Survival']
plt.figure(figsize=(5,5))
plt.pie(diff,labels=['Not Survival','Survival'],autopct='%1.2f%%',colors=['#2A2525'
plt.axis('equal')
plt.title('Distribution of Survival/Non-Survival Rate in Titanic',fontsize=12,fontw
plt.show()
```



Insight : Out of 891 members, The Survival Rate percentage in Titanic is 38.38% and Not survival Rate Percentage is 61.62%

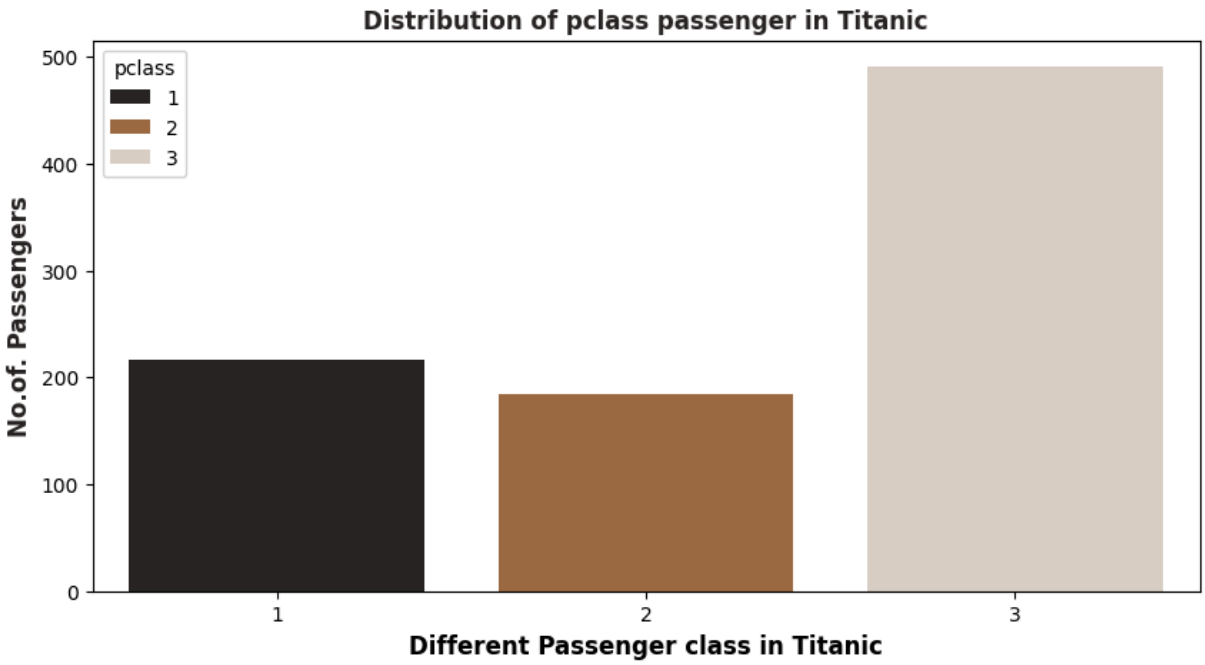
In [ ]:
```
#3.what is distribution of passengers across different classes(Pclass)
df.groupby('pclass')['pclass'].count()
```

Out[ ]:

	pclass
pclass	
1	216
2	184
3	491

dtype: int64

In [ ]:
```
plt.figure(figsize=(10,5))
sns.countplot(x='pclass',data=df,hue='pclass',palette=['#2A2525','#AC6C35','#DBCEBF
plt.title('Distribution of pclass passenger in Titanic',fontsize=12,fontweight='bol
plt.xlabel('Different Passenger class in Titanic',fontsize=12,fontweight='bold',col
plt.ylabel('No.of. Passengers',fontsize=12,fontweight='bold',color='#2A2525')
plt.show()
```



Insight: In Titanic 891 Passenger are travelling in different pclass . 216 member of passenger are travelling in 1st class and 184 member of passenger are travelling in 2nd class and 491 member of passenger are travelling in 3rd class.Most of people are travelling in 3rd class It mean most of the Titanic Passengers are in Middle class .

In [ ]:
```
# Survival for Each class
Survival_1st_class=df[df['pclass']==1]['alive'].value_counts()
Survival_2nd_class=df[df['pclass']==2]['alive'].value_counts()
Survival_3rd_class=df[df['pclass']==3]['alive'].value_counts()
print(f'Survival_1st_class:{Survival_1st_class} \nSurvival_2nd_class:{Survival_2nd_
```

```
Survival_1st_class:alive
yes    136
no      80
Name: count, dtype: int64
Survival_2nd_class:alive
no     97
yes    87
Name: count, dtype: int64
Survival_3rd_class:alive
no     372
yes    119
Name: count, dtype: int64
```

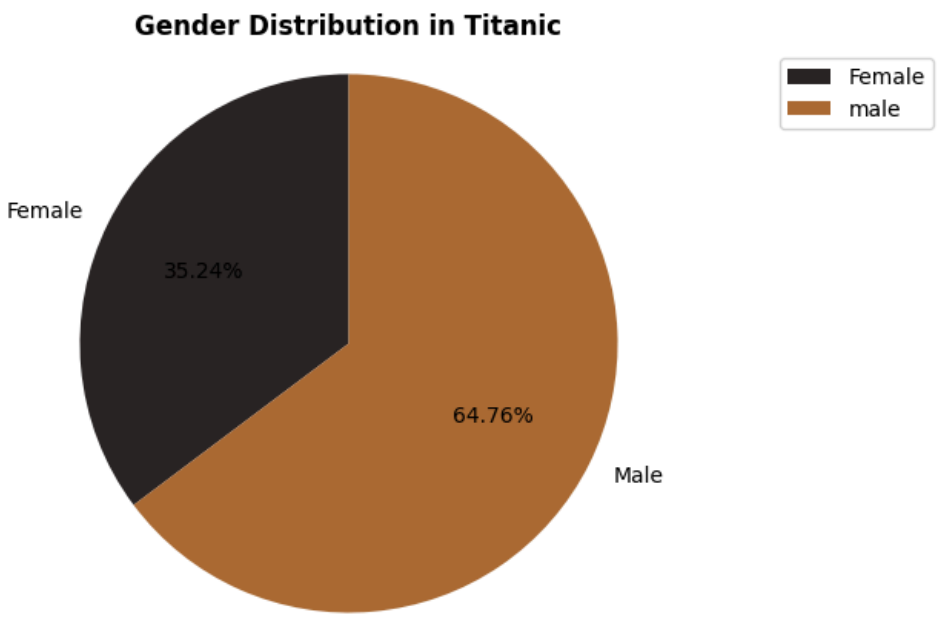
In [ ]:
```
#4. How many male, female Passengers were there?
df['sex'].value_counts()
```

Out[ ]:

sex	count
male	577
female	314

dtype: int64

In [ ]:
```
data=df.groupby('sex')['sex'].value_counts()
Labels=['Female','Male']
plt.figure(figsize=(10,5))
plt.pie(data,labels=Labels,colors=['#2A2525','#AC6C35'],autopct='%1.2f%%',startangl
plt.title('Gender Distribution in Titanic',fontsize=12,fontweight='bold',color='Bla
plt.legend(['Female','male'],loc='upper right')
plt.axis('equal')
plt.show()
```

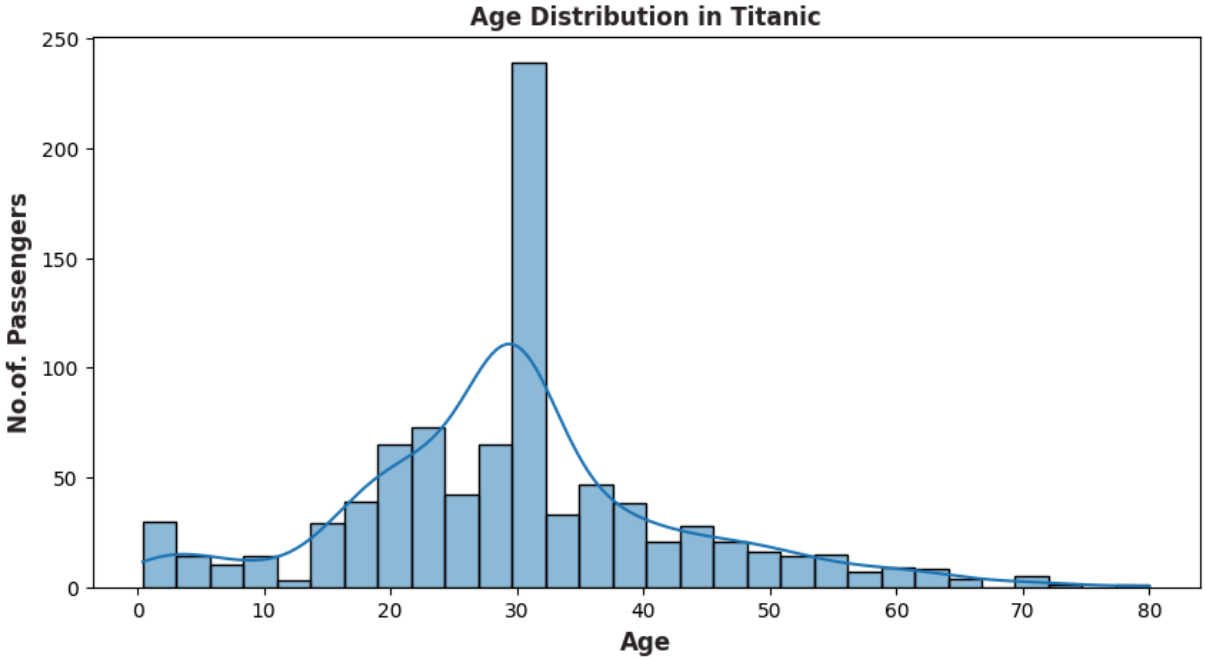


Insight: Gender wise Passenger Percentage are Male is 64.76% and Female is 35.24%. Based on this analysis Most of Male are interest to travel in Titanic boat

In [ ]:
```
#5.what is average age of passengers
df['age'].mean()
```

Out[ ]: np.float64(29.69911764705882)

In [ ]:
```
plt.figure(figsize=(10,5))
sns.histplot(x='age',data=df,kde=True)
plt.title('Age Distribution in Titanic',fontsize=12,fontweight='bold',color='#2A252
plt.xlabel('Age',fontsize=12,fontweight='bold',color='#2A2525')
plt.ylabel('No.of. Passengers',fontsize=12,fontweight='bold',color='#2A2525')
plt.show()
```



Insight: The Average Age of Passenger in Titanic is 29. Most of Younger People are like travel in Titanic.

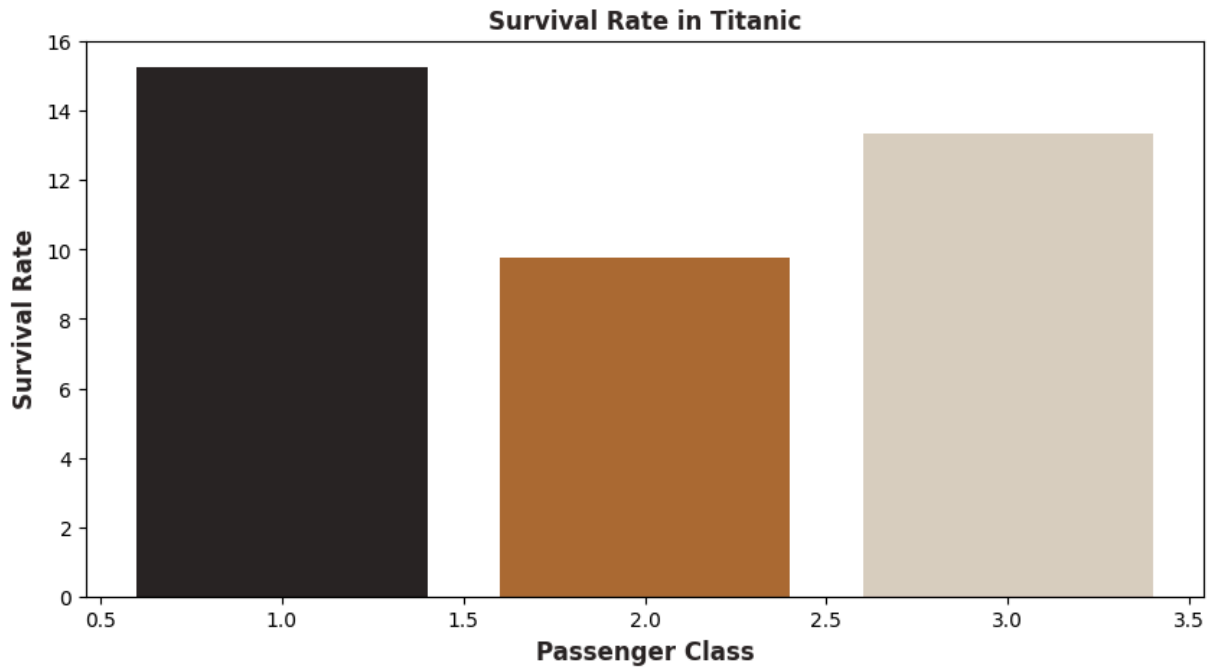
In [ ]:
```
#6.what is survival rate for each clas(Pclass)
passenger=df[df['alive']=='yes'].groupby('pclass')['alive'].count()
Total_Passenger=df['alive'].count()
survival_rate=round(passenger/Total_Passenger*100,2)
survival_rate
```

Out[ ]:

pclass	alive
1	15.26
2	9.76
3	13.36

dtype: float64

In [ ]:
```
plt.figure(figsize=(10,5))
plt.bar(survival_rate.index,survival_rate.values,color=['#2A2525','#AC6C35','#DBCEB
plt.title('Survival Rate in Titanic', fontsize=12,fontweight='bold',color='#2A2525'
plt.xlabel('Passenger Class',fontsize=12,fontweight='bold',color='#2A2525')
plt.ylabel('Survival Rate',fontsize=12,fontweight='bold',color='#2A2525')
plt.show()
```



Insight: Based on the above analysis Most of Survival Passengers are travelling in 1st class(15.26).Least Survival Passengers are travelling in 2nd class(9.76).

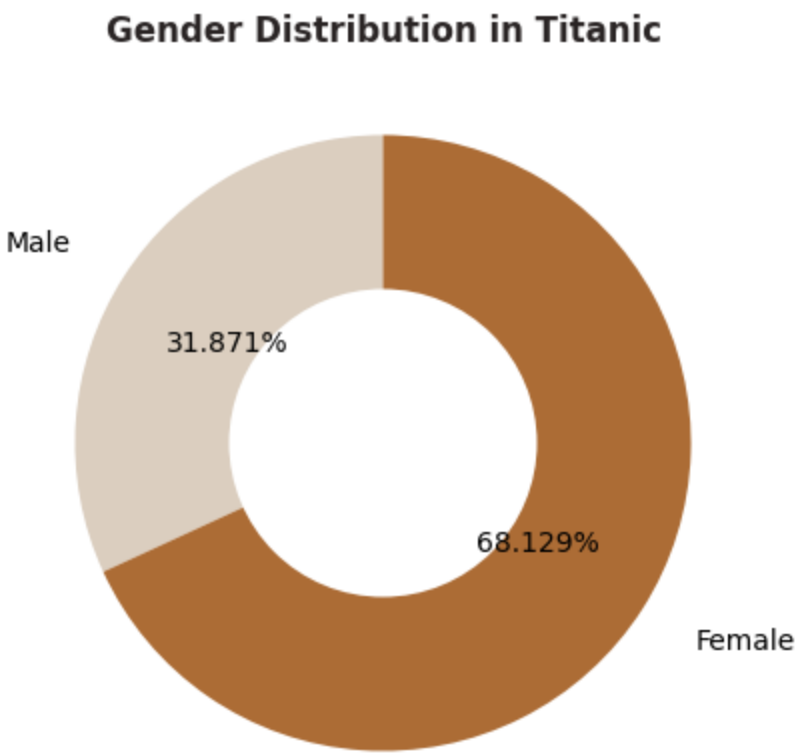
In [ ]:
```
# 7.did woman have a highest survival rate than men?
Total_Passenger=df['sex'].count()
male=df.groupby('sex').apply(lambda x: x[(x['sex']=='male') & (x['alive']=='yes')][
female=df.groupby('sex').apply(lambda x: x[(x['sex']=='female') & (x['alive']=='yes
male_rate=(male/Total_Passenger)*100
female_rate=(female/Total_Passenger)*100
print(f'male_rate:{male_rate} \nfemale_rate:{female_rate}')
```

```
male_rate:12.2334455667789
female_rate:26.15039281705948
```

```
<ipython-input-231-8d8a72f833d8>:10: DeprecationWarning: DataFrameGroupBy.apply oper
ated on the grouping columns. This behavior is deprecated, and in a future version o
f pandas the grouping columns will be excluded from the operation. Either pass `incl
ude_groups=False` to exclude the groupings or explicitly select the grouping columns
after groupby to silence this warning.
  male=df.groupby('sex').apply(lambda x: x[(x['sex']=='male') & (x['alive']=='yes')]
['alive'].count()).loc['male']
<ipython-input-231-8d8a72f833d8>:11: DeprecationWarning: DataFrameGroupBy.apply oper
ated on the grouping columns. This behavior is deprecated, and in a future version o
f pandas the grouping columns will be excluded from the operation. Either pass `incl
ude_groups=False` to exclude the groupings or explicitly select the grouping columns
after groupby to silence this warning.
  female=df.groupby('sex').apply(lambda x: x[(x['sex']=='female') & (x['alive']=='ye
s')]['alive'].count()).loc['female']
```

In [ ]:
```
plt.figure(figsize=(10,5))
plt.pie([male_rate,female_rate],labels=['Male','Female'],autopct='%1.3f%%',colors=[
#Adding a circle at the center
center_circle=plt.Circle((0,0),0.50,fc='white')
fig=plt.gcf()
fig.gca().add_artist(center_circle)
```

```
plt.title('Gender Distribution in Titanic',fontsize=12,fontweight='bold',color='#2A
plt.show()
```



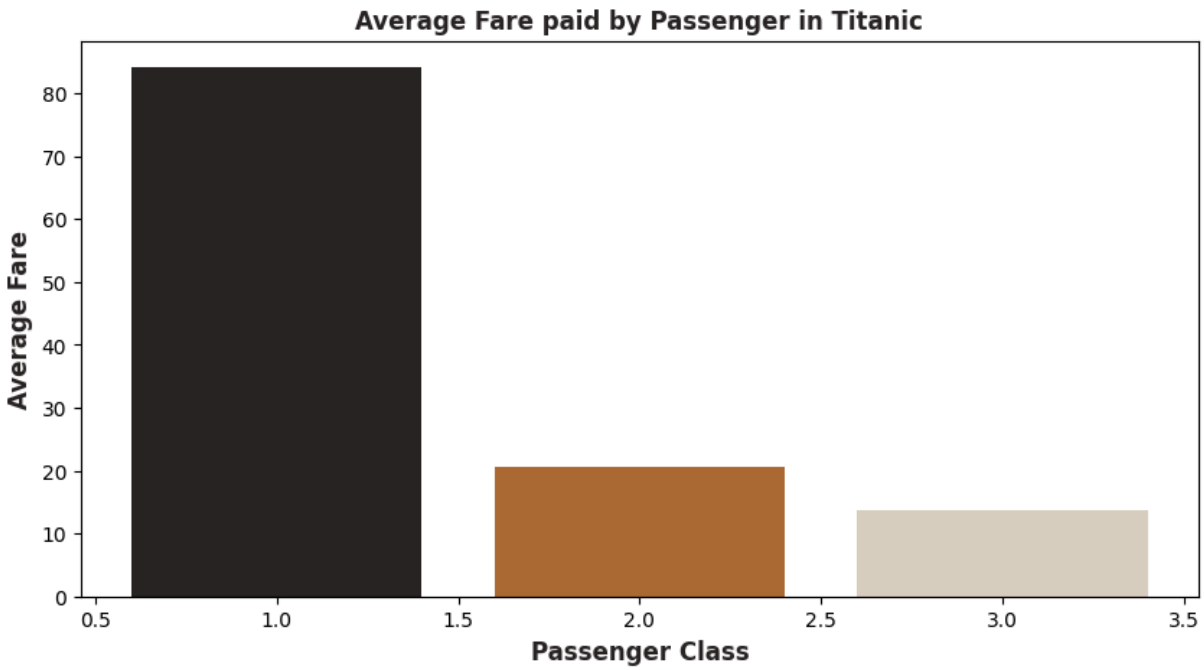
- highest survival rate in Titanic Female with 68.129% and Male Survival Rate in Titanic is 31.871%

In [ ]:
```
#8.what was the average fare paid by passenger in each class?
Fare=df.groupby('pclass')[['pclass','fare']].mean()    #.rename(columns={'fare':'me
Fare
```

Out[ ]:

	pclass	fare
pclass		
1	1.0	84.154687
2	2.0	20.662183
3	3.0	13.675550

In [ ]:
```
plt.figure(figsize=(10,5))
plt.bar(Fare['mean_fare'].index,Fare['mean_fare'].values,color=['#2A2525','#AC6C35'
plt.title('Average Fare paid by Passenger in Titanic',fontsize=12,fontweight='bold'
plt.xlabel('Passenger Class',fontsize=12,fontweight='bold',color='#2A2525')
plt.ylabel('Average Fare',fontweight='bold',fontsize=12,color='#2A2525')
plt.show()
```



Insight:

- Total Fare Mean of 1st class is 84.154
- Total Fare Mean of 2nd class is 20.66
- Total Fare Mean of 3rd class is 13.67

In [ ]:
```
#9.what is correlation between sibsp and survival?
df.groupby('sibsp')[['sibsp','alive']].count()
```

Out[ ]:

	sibsp	alive
sibsp		
0	608	608
1	209	209
2	28	28
3	16	16
4	18	18
5	5	5
8	7	7

CONCLUSION:

In this Titanic data analysis project, we conducted comprehensive exploratory data analysis on the Titanic dataset using Python. The dataset contained information on 891 passengers,

including demographic and travel-related details. Through the analysis, we derived the following key insights:

- **Survival Rate:** Only 38.38% of passengers survived the disaster, while 61.62% did not survive.
- **Passenger Class Impact:** Most passengers (55%) were in 3rd class. However, survival rates were highest in 1st class (15.26%) and lowest in 2nd class (9.76%), highlighting that passenger class significantly influenced chances of survival.
- **Gender Distribution & Survival:** 64.76% of the passengers were male, and 35.24% were female. Females had a higher survival rate than males, suggesting gender played a key role during evacuation and rescue.
- **Average_Age Factor:** The average age of passengers was approximately 29 years, indicating most travelers were young adults. There was no strong correlation between age and survival based on this data alone.
- **Embarkation & Deck:** Majority of passengers embarked from Southampton, and the most common deck recorded was 'C'. Missing values were appropriately handled to ensure data integrity.

This analysis highlights the influence of socio-economic status, gender, and class on survival during the Titanic tragedy. Such insights not only help understand historical events but also demonstrate the value of data analysis in uncovering patterns and informing future safety measures.