In [1]:

```
import seaborn as sns
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:

```
dataset=sns.load_dataset('titanic')
dataset
```

Out[2]:

	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	F
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	F
3	1	1	female	35.0	1	0	53.1000	S	First	woman	F
4	0	3	male	35.0	0	0	8.0500	S	Third	man	T
...	...	...	...	...	...	...	...	...	...	...	
886	0	2	male	27.0	0	0	13.0000	S	Second	man	T
887	1	1	female	19.0	0	0	30.0000	S	First	woman	F
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	F
889	1	1	male	26.0	0	0	30.0000	C	First	man	T
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	T

891 rows × 15 columns

In [3]:

```
dataset.isnull().sum()
```

Out[3]:

```
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
```

In [4]:

```
dataset['age']=dataset["age"].ffill()
dataset['age']
```

Out[4]:

```
0      22.0
1      38.0
2      26.0
3      35.0
4      35.0
       ...
886    27.0
887    19.0
888    19.0
889    26.0
890    32.0
Name: age, Length: 891, dtype: float64
```

In [5]:

```
dataset['deck']=dataset["deck"].bfill()
dataset['deck']
```

Out[5]:

```
0        C
1        C
2        C
3        C
4        E
      ...
886      B
887      B
888      C
889      C
890    NaN
Name: deck, Length: 891, dtype: category
Categories (7, object): ['A', 'B', 'C', 'D', 'E', 'F', 'G']
```

In [6]:

```
dataset['embarked']=dataset["embarked"].bfill()
dataset['embarked']
```

Out[6]:

```
0      S
1      C
2      S
3      S
4      S
      ..
886    S
887    S
888    S
889    C
890    Q
Name: embarked, Length: 891, dtype: object
```

In [7]:

```
dataset['embark_town']=dataset["embark_town"].bfill()
dataset['embark_town']
```

Out[7]:

```
0      Southampton
1        Cherbourg
2      Southampton
3      Southampton
4      Southampton
          ...
886    Southampton
887    Southampton
888    Southampton
889      Cherbourg
890     Queenstown
Name: embark_town, Length: 891, dtype: object
```

In [8]:

```
dataset
```

Out[8]:

	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
...	...	...	...	...	...	...	...	...	...	...	
886	0	2	male	27.0	0	0	13.0000	S	Second	man	T
887	1	1	female	19.0	0	0	30.0000	S	First	woman	Fa
888	0	3	female	19.0	1	2	23.4500	S	Third	woman	Fa
889	1	1	male	26.0	0	0	30.0000	C	First	man	T
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	T

891 rows × 15 columns

In [9]:

```
dataset.isnull().sum()
```

Out[9]:

```
survived       0
pclass         0
sex            0
age            0
sibsp          0
parch          0
fare           0
embarked       0
class          0
who            0
adult_male     0
deck           1
embark_town    0
alive          0
alone          0
dtype: int64
```

1) Find the female those who are not alone with fare value is less than 60000and greater then 20000.

In [10]:

```
data1=dataset.iloc[:,[2,6,14]]
data1
```

Out[10]:

	sex	fare	alone
0	male	7.2500	False
1	female	71.2833	False
2	female	7.9250	True
3	female	53.1000	False
4	male	8.0500	True
...	...	...	...
886	male	13.0000	True
887	female	30.0000	True
888	female	23.4500	False
889	male	30.0000	True
890	male	7.7500	True

891 rows × 3 columns

In [11]:

```
FARE=[]
for x in data1["fare"]:
    if x>=20 and x<60:
        FARE.append(0)
    else:
        FARE.append(1)
print(FARE)
```

```
[1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0,
0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1,
0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0,
1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0,
1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1,
1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,
1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1,
1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,
1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1,
1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0,
1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1,
1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0,
1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0,
0, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1,
0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1,
1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0,
1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 1,
1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0,
0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1,
0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1,
1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1,
1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1,
1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0,
1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1,
0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0,
1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0,
1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0,
1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,
0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0,
1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1]
```

In [12]:

```
data1.insert(2,"FARE",FARE)
```

In [13]:

```
del data1["fare"]
```

In [14]:

```
data1
```

Out[14]:

	sex	FARE	alone
0	male	1	False
1	female	1	False
2	female	1	True
3	female	0	False
4	male	1	True
...	...	...	...
886	male	1	True
887	female	0	True
888	female	0	False
889	male	0	True
890	male	1	True

891 rows × 3 columns

In [15]:

```
data2=data1.loc[(data1["sex"]=="female")&(data1["FARE"]==0)]
data2
```

Out[15]:

	sex	FARE	alone
3	female	0	False
9	female	0	False
11	female	0	True
24	female	0	False
25	female	0	False
...	...	...	...
874	female	0	False
880	female	0	False
885	female	0	False
887	female	0	True
888	female	0	False

100 rows × 3 columns

In [16]:

```
from sklearn.preprocessing import LabelEncoder
l1=LabelEncoder()
data2["alone"]=l1.fit_transform(data2["alone"])
data2["alone"]
```

```
C:\Users\sujat\AppData\Local\Temp\ipykernel_10044\2886656297.py:3: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  data2["alone"]=l1.fit_transform(data2["alone"])
```

Out[16]:

```
3      0
9      0
11     1
24     0
25     0
      ..
874    0
880    0
885    0
887    1
888    0
Name: alone, Length: 100, dtype: int64
```

In [17]:

```
data3=data2.loc[(data2["alone"]==0)]
data3
```

Out[17]:

	sex	FARE	alone
3	female	0	0
9	female	0	0
24	female	0	0
25	female	0	0
41	female	0	0
...	...	...	...
871	female	0	0
874	female	0	0
880	female	0	0
885	female	0	0
888	female	0	0

88 rows × 3 columns

In [18]:

```
data2["sex"].unique()
```

Out[18]:

```
array(['female'], dtype=object)
```

In [19]:

```
data2["alone"].unique()
```

Out[19]:

```
array([0, 1], dtype=int64)
```

In [20]:

```
data2["FARE"].unique()
```

Out[20]:

```
array([0], dtype=int64)
```

2) Find the Third class travled people and find the count of teenage and adult age and old age.

In [48]:

```
dataset
```

Out[48]:

	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
...	...	...	...	...	...	...	...	...	...	...	
886	0	2	male	27.0	0	0	13.0000	S	Second	man	T
887	1	1	female	19.0	0	0	30.0000	S	First	woman	Fa
888	0	3	female	19.0	1	2	23.4500	S	Third	woman	Fa
889	1	1	male	26.0	0	0	30.0000	C	First	man	T
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	T

891 rows × 15 columns

In [49]:

```
data1=dataset.iloc[:,[8,3]]
data1
```

Out[49]:

	class	age
0	Third	22.0
1	First	38.0
2	Third	26.0
3	First	35.0
4	Third	35.0
...	...	...
886	Second	27.0
887	First	19.0
888	Third	19.0
889	First	26.0
890	Third	32.0

891 rows × 2 columns

In [50]:

```
data1["age"].value_counts()
```

Out[50]:

```
24.0    39
19.0    38
21.0    33
22.0    32
28.0    32
        ..
70.5     1
36.5     1
55.5     1
20.5     1
74.0     1
Name: age, Length: 88, dtype: int64
```

In [51]:

```
AGE=[]
for x in data1["age"]:
    if x>0 and x<=15:
        AGE.append("child")
    elif x>15 and x<=30:
        AGE.append("teenage")
    elif x>30 and x<=50:
        AGE.append("adult")
    else:
        AGE.append("old age")
print(AGE)
```

```
['teenage', 'adult', 'teenage', 'adult', 'adult', 'adult', 'old age', 'chi
ld', 'teenage', 'child', 'child', 'old age', 'teenage', 'adult', 'child',
'old age', 'child', 'child', 'adult', 'adult', 'adult', 'adult', 'child',
'teenage', 'child', 'adult', 'adult', 'teenage', 'teenage', 'teenage', 'ad
ult', 'adult', 'adult', 'old age', 'teenage', 'adult', 'adult', 'teenage',
'teenage', 'child', 'adult', 'teenage', 'teenage', 'child', 'teenage', 'te
enage', 'teenage', 'teenage', 'teenage', 'teenage', 'child', 'teenage', 'a
dult', 'teenage', 'old age', 'old age', 'teenage', 'teenage', 'child', 'ch
ild', 'teenage', 'adult', 'adult', 'child', 'child', 'child', 'teenage',
'teenage', 'teenage', 'teenage', 'adult', 'teenage', 'teenage', 'teenage',
'adult', 'teenage', 'teenage', 'teenage', 'child', 'teenage', 'teenage',
'teenage', 'teenage', 'teenage', 'teenage', 'adult', 'teenage', 'teenage',
'teenage', 'teenage', 'teenage', 'teenage', 'adult', 'teenage', 'old age',
'old age', 'old age', 'teenage', 'adult', 'adult', 'teenage', 'teenage',
'teenage', 'adult', 'adult', 'teenage', 'teenage', 'teenage', 'adult', 'ad
ult', 'adult', 'child', 'teenage', 'teenage', 'teenage', 'teenage', 'old a
ge', 'teenage', 'teenage', 'child', 'teenage', 'teenage', 'adult', 'adul
t', 'old age', 'child', 'child', 'teenage', 'teenage', 'adult', 'adult',
'teenage', 'adult', 'teenage', 'teenage', 'teenage', 'teenage', 'adult',
```

In [52]:

```
data1.insert(0,"AGE",AGE)
```

In [53]:

```
del data1["age"]
```

In [54]:

```
data1
```

Out[54]:

	AGE	class
0	teenage	Third
1	adult	First
2	teenage	Third
3	adult	First
4	adult	Third
...	...	...
886	teenage	Second
887	teenage	First
888	teenage	Third
889	teenage	First
890	adult	Third

891 rows × 2 columns

In [55]:

```
data2=data1.loc[(data1["class"]=="Third")&(data1["AGE"]!="child")]
data2
```

Out[55]:

	AGE	class
0	teenage	Third
2	teenage	Third
4	adult	Third
5	adult	Third
8	teenage	Third
...	...	...
882	teenage	Third
884	teenage	Third
885	adult	Third
888	teenage	Third
890	adult	Third

419 rows × 2 columns

In [56]:

```
data2["AGE"].value_counts()
```

Out[56]:

```
teenage    257
adult      142
old age     20
Name: AGE, dtype: int64
```

In [57]:

```
data2["class"].unique()
```

Out[57]:

```
['Third']
Categories (3, object): ['First', 'Second', 'Third']
```

In [58]:

```
data2["AGE"].unique()
```

Out[58]:

```
array(['teenage', 'adult', 'old age'], dtype=object)
```

3) Find the count of teenage and adult ade and old age.

In [30]:

```
data1
```

Out[30]:

	AGE	class
0	teenage	Third
1	adult	First
2	teenage	Third
3	adult	First
4	adult	Third
...	...	...
886	teenage	Second
887	teenage	First
888	teenage	Third
889	teenage	First
890	adult	Third

891 rows × 2 columns

In [31]:

```
data1["AGE"].value_counts()
```

Out[31]:

```
teenage    401
adult      307
child      106
old age     77
Name: AGE, dtype: int64
```

4) Find the Alive Man and chlids from 1st and second class with fare of above 25000.

In [32]:

```
dataset
```

Out[32]:

	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
...	...	...	...	...	...	...	...	...	...	...	
886	0	2	male	27.0	0	0	13.0000	S	Second	man	T
887	1	1	female	19.0	0	0	30.0000	S	First	woman	Fa
888	0	3	female	19.0	1	2	23.4500	S	Third	woman	Fa
889	1	1	male	26.0	0	0	30.0000	C	First	man	T
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	T

891 rows × 15 columns

In [33]:

```
data1=dataset.iloc[:,[6,8,9,13]]
data1
```

Out[33]:

	fare	class	who	alive
0	7.2500	Third	man	no
1	71.2833	First	woman	yes
2	7.9250	Third	woman	yes
3	53.1000	First	woman	yes
4	8.0500	Third	man	no
...	...	...	...	...
886	13.0000	Second	man	no
887	30.0000	First	woman	yes
888	23.4500	Third	woman	no
889	30.0000	First	man	yes
890	7.7500	Third	man	no

891 rows × 4 columns

In [34]:

```
data2=data1.loc[(data1["alive"]=="yes")]
data2
```

Out[34]:

	fare	class	who	alive
1	71.2833	First	woman	yes
2	7.9250	Third	woman	yes
3	53.1000	First	woman	yes
8	11.1333	Third	woman	yes
9	30.0708	Second	child	yes
...	...	...	...	...
875	7.2250	Third	child	yes
879	83.1583	First	woman	yes
880	26.0000	Second	woman	yes
887	30.0000	First	woman	yes
889	30.0000	First	man	yes

342 rows × 4 columns

In [35]:

```
data3=data2.loc[(data2["who"]=="man")|(data2["who"]=="child")]
data3
```

Out[35]:

	fare	class	who	alive
9	30.0708	Second	child	yes
10	16.7000	Third	child	yes
17	13.0000	Second	man	yes
21	13.0000	Second	man	yes
22	8.0292	Third	child	yes
...	...	...	...	...
839	29.7000	First	man	yes
857	26.5500	First	man	yes
869	11.1333	Third	child	yes
875	7.2250	Third	child	yes
889	30.0000	First	man	yes

137 rows × 4 columns

In [36]:

```
data4=data3.loc[(data3["class"]=="First")|(data2["class"]=="Second")]
data4
```

Out[36]:

	fare	class	who	alive
9	30.0708	Second	child	yes
17	13.0000	Second	man	yes
21	13.0000	Second	man	yes
23	35.5000	First	man	yes
43	41.5792	Second	child	yes
...	...	...	...	...
827	37.0042	Second	child	yes
831	18.7500	Second	child	yes
839	29.7000	First	man	yes
857	26.5500	First	man	yes
889	30.0000	First	man	yes

74 rows × 4 columns

In [37]:

```
data5=data4.loc[(data4["fare"]>=25)]
data5
```

Out[37]:

	fare	class	who	alive
9	30.0708	Second	child	yes
23	35.5000	First	man	yes
43	41.5792	Second	child	yes
55	35.5000	First	man	yes
58	27.7500	Second	child	yes
...	...	...	...	...
802	120.0000	First	child	yes
827	37.0042	Second	child	yes
839	29.7000	First	man	yes
857	26.5500	First	man	yes
889	30.0000	First	man	yes

62 rows × 4 columns

In [38]:

```
data5["fare"].value_counts()
```

Out[38]:

```
26.5500     7
26.0000     4
30.5000     4
26.2875     3
30.0000     3
35.5000     3
120.0000    3
26.2500     2
512.3292    2
39.0000     2
76.7292     2
52.5542     2
56.9292     1
133.6500    1
30.0708     1
26.3875     1
211.3375    1
57.0000     1
52.0000     1
33.0000     1
53.1000     1
37.0042     1
79.2000     1
89.1042     1
110.8833    1
36.7500     1
91.0792     1
81.8583     1
55.4417     1
151.5500    1
90.0000     1
31.0000     1
63.3583     1
29.0000     1
27.7500     1
41.5792     1
29.7000     1
Name: fare, dtype: int64
```

In [39]:

```
data5["class"].unique()
```

Out[39]:

```
['Second', 'First']
Categories (3, object): ['First', 'Second', 'Third']
```

In [40]:

```
data5["who"].unique()
```

Out[40]:

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

In [41]:

```
data5["alive"].unique()
```

Out[41]:

```
array(['yes'], dtype=object)
```

In [ ]:

```

```