```
import pandas as pd
import numpy as np
```

```
data = {
    'name': ['Alice','Bob','Charlie','David','Eve'],
    'Age' : [25,30,35,40,None],
    'Gender' : ['female','male','male','male','female'],
    'Salary' : [50000,60000,70000,80000,90000]
}
```

```
df = pd.DataFrame(data)
```

```
df
```

	name	Age	Gender	Salary
0	Alice	25.0	female	50000
1	Bob	30.0	male	60000
2	Charlie	35.0	male	70000
3	David	40.0	male	80000
4	Eve	NaN	female	90000

```
import pandas as pd
import numpy as np
```

```
df1 = pd.DataFrame(data=np.random.randint(0,30,(6,4)),
               columns = ['co11','co12','col3','col4'],
               index = ['A','B','C','D','E','F'])
```

```
df1
```

	co11	co12	col3	col4
A	18	12	15	6
B	21	18	25	21
C	10	10	21	18
D	20	24	14	28
E	15	14	20	19
F	10	18	15	27

```
df1.shape
```

```
(6, 4)
```

```
df
```

	name	Age	Gender	Salary
0	Alice	25.0	female	50000
1	Bob	30.0	male	60000
2	Charlie	35.0	male	70000
3	David	40.0	male	80000
4	Eve	NaN	female	90000

```
df.head(2)
```

	name	Age	Gender	Salary
0	Alice	25.0	female	50000
1	Bob	30.0	male	60000

```
df.tail()
```

	name	Age	Gender	Salary
0	Alice	25.0	female	50000
1	Bob	30.0	male	60000
2	Charlie	35.0	male	70000
3	David	40.0	male	80000
4	Eve	NaN	female	90000

```
df.describe()
```

	Age	Salary
count	4.000000	5.000000
mean	32.500000	70000.000000
std	6.454972	15811.388301
min	25.000000	50000.000000
25%	28.750000	60000.000000
50%	32.500000	70000.000000
75%	36.250000	80000.000000
max	40.000000	90000.000000

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   name    5 non-null      object
 1   Age     4 non-null      float64
 2   Gender  5 non-null      object
 3   Salary  5 non-null      int64
dtypes: float64(1), int64(1), object(2)
memory usage: 292.0+ bytes
```

```
df1.tail(2)
```

	co11	co12	col3	col4
E	15	14	20	19
F	10	18	15	27

```
df[['Age','Salary']].mean()
```

	0
Age	32.5
Salary	70000.0

dtype: float64

```
df[['Age','Salary']].max()
```

	0
Age	40.0
Salary	90000.0

dtype: float64

```
df[['Age','Salary']].min()
```

	0
Age	25.0
Salary	50000.0

dtype: float64

```
df['Salary'].value_counts()
```

Salary	count
50000	1
60000	1
70000	1
80000	1
90000	1

dtype: int64

```
df1.count()
```

	0
co11	6
co12	6
col3	6
col4	6

dtype: int64

```
df.count()
```

	0
name	5
Age	5
Gender	5
Salary	5

dtype: int64

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   name    5 non-null      object
 1   Age     5 non-null      object
 2   Gender  5 non-null      object
 3   Salary  5 non-null      int64
dtypes: int64(1), object(3)
memory usage: 292.0+ bytes
```

```
df.groupby('Salary').sum()
```

Salary	name	Age	Gender
50000	Alice	25	female
60000	Bob	30	male
70000	Charlie	35	male
80000	David	40	male
90000	Eve	none	female

```
df[['Age','Salary']].sum()
```

	0
Age	130.0
Salary	350000.0

dtype: float64

```
import numpy as np
import pandas as pd
```

Start coding or generate with AI.

```
from datetime import datetime
def age_calculation(Birthday):
   Birth_day = datetime.strptime(Birthday, "%d-%m-%Y")
   current_date= datetime.now()
   age= current_date.year - Birth_day.year - ((current_date.month, current_date.day) < (Birth_day.month, Birth_day.day))
   return age
def convert_rupee(salary_rupees):
  convertion = 0.012
  return salary_rupees*convertion
salary_rupees = float(input())
Birthday = input()
salary_dollar = convert_rupee(salary_rupees)
age= age_calculation(Birthday)
```

```
100000
27-08-1970
```

```
salary_dollar
```

```
1200.0
```

```
age
```

```
54
```

```
def reverse_num(n) :
  return int(str(n)[::-1])
n= int(input())
print(reverse_num(n))
```

```
123
321
```

```
n= int(input())
x=n
while x!=0:
  digit = x%10
  x = x//10
  print(digit,end="")
```

```
123
321
```

```python
n= int(input())
for i in range(n-1):
  for j in range(n):
    print("*", end='')
  print()
```

```
10
**********
**********
**********
**********
**********
**********
**********
**********
**********
```

Start coding or generate with AI.

```
import numpy as np
arr1 = np.array([10,20,30,40,50])
print(arr1)
```

```
[10 20 30 40 50]
```

```
arr1.ndim
```

```
1
```

```
import numpy as np
arr2 = np.array([[10,20,30],[40,50,60]])
arr2
```

```
array([[10, 20, 30],
       [40, 50, 60]])
```

```
arr1[-1]
```

```
np.int64(50)
```

```
arr1[-2:]
```

```
array([40, 50])
```

```
arr1[::-1]
```

```
array([50, 40, 30, 20, 10])
```

```
arr1[-2:]
```

```
array([40, 50])
```

```
arr1[-5:]
```

```
array([10, 20, 30, 40, 50])
```

```
B = np.arange(10,22,2)
B
```

```
array([10, 12, 14, 16, 18, 20])
```

```
arr3 = np.random.randint(1,5,(4,4))
```

```
arr3
```

```
array([[3, 1, 2, 4],
       [1, 2, 2, 2],
       [2, 4, 1, 4],
       [4, 2, 1, 1]])
```

```
c = np.linspace(10,2,6)
c
```

```
array([10. ,  8.4,  6.8,  5.2,  3.6,  2. ])
```

```
A = np.array([3.14,11.2,0.5,1.62])
A
```

```
array([ 3.14, 11.2 ,  0.5 ,  1.62])
```

```
type(A)
```

```
numpy.ndarray
```

```
A.dtype
```

```
dtype('float64')
```

```
A.size
```

```
4
```

```
A.shape
```

```
(4,)
```

```
A.ndim
```

```
1
```

```
arr3.shape
```

```
(4, 4)
```

```
arr2.ndim
```

```
2
```

```
arr4 = arr3.reshape(4,4)
arr4
```

```
array([[3, 1, 2, 4],
       [1, 2, 2, 2],
       [2, 4, 1, 4],
       [4, 2, 1, 1]])
```

```
arr4
```

```
array([[3, 1, 2, 4],
       [1, 2, 2, 2],
       [2, 4, 1, 4],
       [4, 2, 1, 1]])
```

```
arr3.reshape(4,4)
```

```
array([[3, 1, 2, 4],
       [1, 2, 2, 2],
       [2, 4, 1, 4],
       [4, 2, 1, 1]])
```

```
arr3[::-1,:]
```

```
array([[4, 2, 1, 1],
       [2, 4, 1, 4],
       [1, 2, 2, 2],
       [3, 1, 2, 4]])
```

```
arr3
```

```
array([[3, 1, 2, 4],
       [1, 2, 2, 2],
       [2, 4, 1, 4],
       [4, 2, 1, 1]])
```

```
A = np.array([[1,2,3],[4,5,6]])
B = np.array([1,2,3])
C = A + B
```

```
C
```

```
array([[2, 4, 6],
       [5, 7, 9]])
```

```
A = np.array([[1,2,3],[4,5,6]])
B = np.array([[7,8,9],[10,11,12]])
```

```
c1 = np.add(A,B)
c2 = np.subtract(A,B)
c3 = np.multiply(A,B)
c4 = np.transpose(A)
```

```
c1
```

```
array([[ 8, 10, 12],
       [14, 16, 18]])
```

```
c2
```

```
array([[-6, -6, -6],
       [-6, -6, -6]])
```

```
c3
```

```
array([[ 7, 16, 27],
       [40, 55, 72]])
```

```
c4
```

```
array([[1, 4],
       [2, 5],
       [3, 6]])
```

```
A
```

```
array([[1, 2, 3],
       [4, 5, 6]])
```

```
B
```

```
array([[ 7,  8,  9],
       [10, 11, 12]])
```

```
C = np.array([[1,2,3],[4,5,6]])
D = np.array([[7,8,9]])
```

```
print("hello")
```

```
hello
```

```
horizontal= np.hstack((A,B))
horizontal
```

```
array([[ 1,  2,  3,  7,  8,  9],
       [ 4,  5,  6, 10, 11, 12]])
```

```
vertical = np.vstack((A,B))
vertical
```

```
array([[ 1,  2,  3],
       [ 4,  5,  6],
       [ 7,  8,  9],
       [10, 11, 12]])
```

```
import numpy as np
arr5 = np.array([[624,77,85,70,83,82],[625,71,72,73,74,75],[626,60,90,92,98,87],[627,88,86,43,95,97]])
```

```
arr5
```

```
array([[624,  77,  85,  70,  83,  82],
       [625,  71,  72,  73,  74,  75],
       [626,  60,  90,  92,  98,  87],
       [627,  88,  86,  43,  95,  97]])
```

```
(r,c) = arr5.shape
print("Total Students:",r)
print("Total Students:",c)
```

```
Total Students: 4
Total Students: 6
```

```
print("All the stuident roll no : ",arr5[::,0])
```

```
All the stuident roll no :  [624 625 626 627]
```

```
print("subject 1 marks",arr5[::,1])
```

```
subject 1 marks [77 71 60 88]
```

```
print("Min marks in subject",np.min(arr5[::,2]))
```

```
Min marks in subject 72
```

```
print("Max marks in subject",np.max([arr5[::,2]]))
```

```
Max marks in subject 90
```

```
print("All subject of the marks:",arr5[::,1::])
```

```
All subject of the marks: [[77 85 70 83 82]
 [71 72 73 74 75]
```

```
 [60 90 92 98 87]
 [88 86 43 95 97]]
```

```
print("Total marks",np.sum(arr5[::,1::],axis=1))
```

```
Total marks [397 365 427 409]
```

```
avg = np.mean(arr5[::,1::],axis=1)
print(np.round(avg,1))
print(np.round(avg))
```

```
[79.4 73.  85.4 81.8]
[79. 73. 85. 82.]
```

```
print("average marks of each subject ",np.mean(arr5[::,1::],axis=0))
```

```
average marks of each subject  [74.   83.25 69.5  87.5  85.25]
```

Start coding or generate with AI.

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

```
!pip install seaborn
```

```
Requirement already satisfied: seaborn in /usr/local/lib/python3.11/dist-packages (0.13.2)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /usr/local/lib/python3.11/dist-packages (from seaborn) (2.0.2)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.11/dist-packages (from seaborn) (2.2.2)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /usr/local/lib/python3.11/dist-packages (from seaborn) (3.10.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (4.57.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (24.2)
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (11.2.1)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.2->seaborn) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.2->seaborn) (2025.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.17.0)
```

```
df1 = pd.read_csv('titanic (1).csv')
```

```
df1.shape
```

```
(891, 15)
```

```
df1.columns
```

```
Index(['survived', 'pclass', 'sex', 'age', 'sibsp', 'parch', 'fare',
       'embarked', 'class', 'who', 'adult_male', 'deck', 'embark_town',
       'alive', 'alone'],
      dtype='object')
```

```
df1.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    object
 9   who          891 non-null    object
 10  adult_male   891 non-null    bool
 11  deck         203 non-null    object
 12  embark_town  889 non-null    object
 13  alive        891 non-null    object
 14  alone        891 non-null    bool
dtypes: bool(2), float64(2), int64(4), object(7)
memory usage: 92.4+ KB
```

```
df1.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

```
df1.tail()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
886	0	2	male	27.0	0	0	13.00	S	Second	man	True	NaN	Southampton	no	True
887	1	1	female	19.0	0	0	30.00	S	First	woman	False	B	Southampton	yes	True
888	0	3	female	NaN	1	2	23.45	S	Third	woman	False	NaN	Southampton	no	False
889	1	1	male	26.0	0	0	30.00	C	First	man	True	C	Cherbourg	yes	True
890	0	3	male	32.0	0	0	7.75	Q	Third	man	True	NaN	Queenstown	no	True

```
df1.tail(10)
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
881	0	3	male	33.0	0	0	7.8958	S	Third	man	True	NaN	Southampton	no	True
882	0	3	female	22.0	0	0	10.5167	S	Third	woman	False	NaN	Southampton	no	True
883	0	2	male	28.0	0	0	10.5000	S	Second	man	True	NaN	Southampton	no	True
884	0	3	male	25.0	0	0	7.0500	S	Third	man	True	NaN	Southampton	no	True
885	0	3	female	39.0	0	5	29.1250	Q	Third	woman	False	NaN	Queenstown	no	False
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True	NaN	Southampton	no	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False	B	Southampton	yes	True
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False	NaN	Southampton	no	False
889	1	1	male	26.0	0	0	30.0000	C	First	man	True	C	Cherbourg	yes	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True	NaN	Queenstown	no	True

Start coding or generate with AI.

```
df1.describe()
```

	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

```
df1.isnull().sum()
```

	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

```
df1.isnull().count()
```

	0
survived	891
pclass	891
sex	891
age	891
sibsp	891
parch	891
fare	891
embarked	891
class	891
who	891
adult_male	891
deck	891
embark_town	891
alive	891
alone	891

dtype: int64

```
df1.isna().sum()
```

	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

```
average = df1['age'].mean()

df1['age'] = df1['age'].fillna(average)
```

```
df1.groupby('sibsp')['survived'].mean()
```

sibsp	survived
0	0.345395
1	0.535885
2	0.464286
3	0.250000
4	0.166667
5	0.000000
8	0.000000

dtype: float64

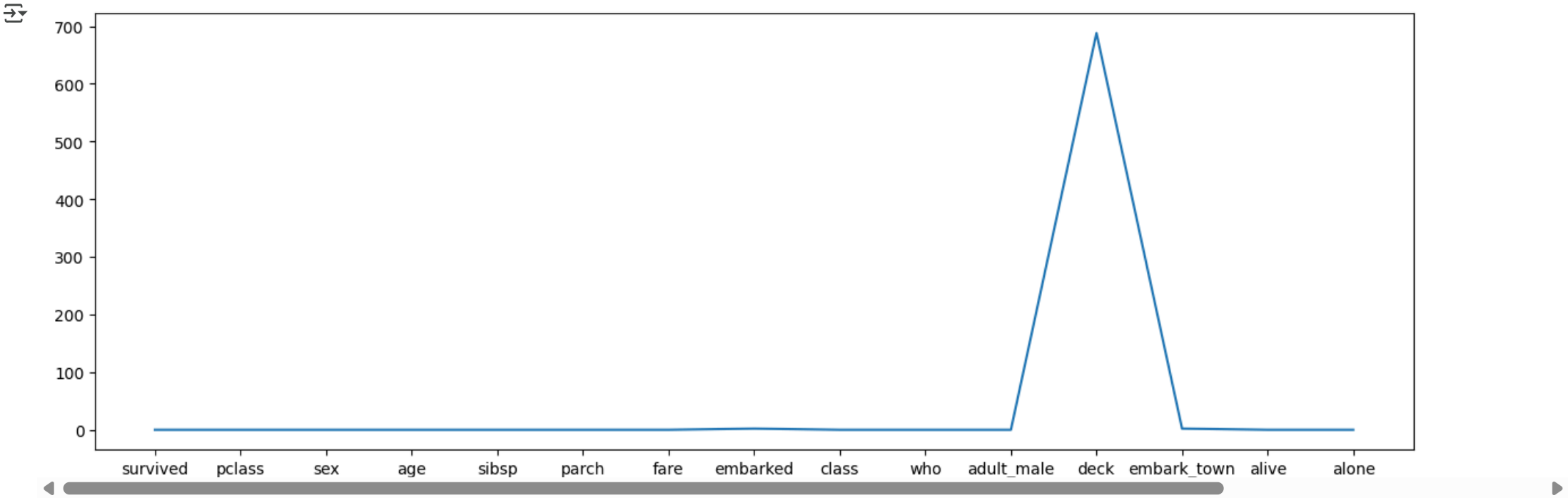
```
import warnings
warnings.filterwarnings('ignore')
```

```
df1.isna().sum()
```

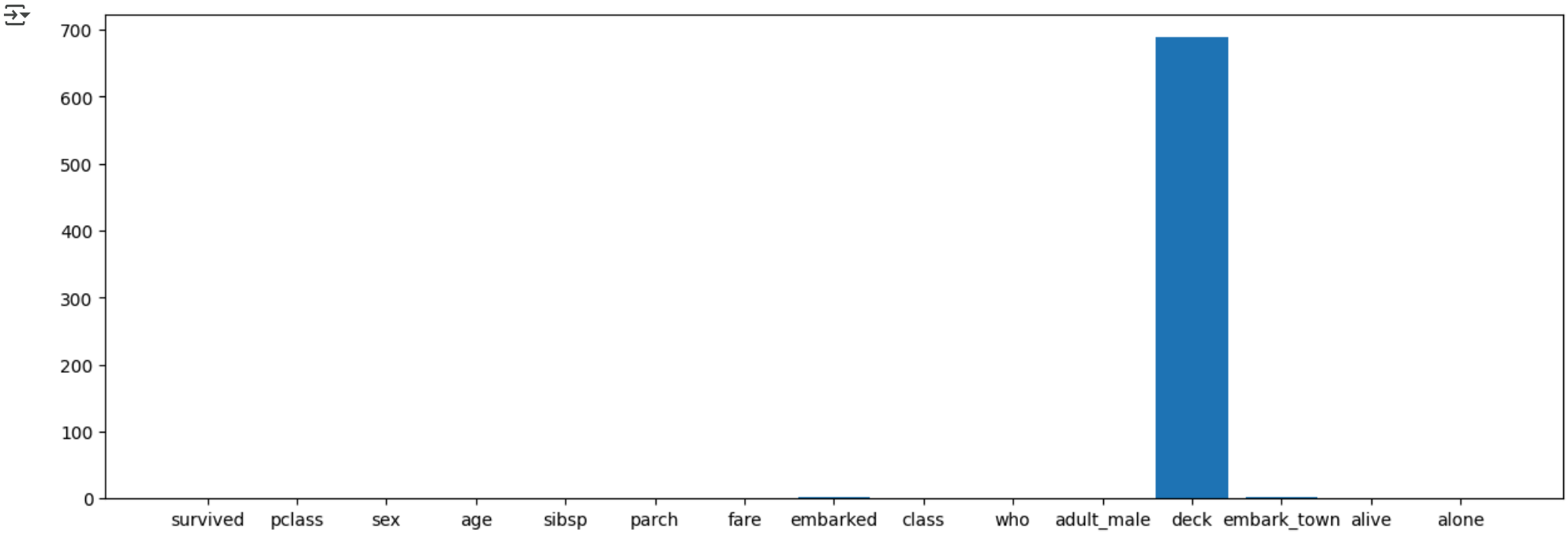
	0
survived	0
pclass	0
sex	0
age	0
sibsp	0
parch	0
fare	0
embarked	2
class	0
who	0
adult_male	0
deck	688
embark_town	2
alive	0
alone	0

dtype: int64

```
plt.figure(figsize = (15,5))
plt.plot(df1.columns,df1.isna().sum())
plt.show()
```




```
plt.figure(figsize = (15,5))
plt.bar(df1.columns,df1.isna().sum())
plt.show()
```



```
plt.figure(figsize = (15,5))
plt.bar(df1.columns,df1.isna().sum(),color = 'yellow')
plt.show()
```


```
sb.countplot(x ='survived',data = df1)
plt.show()
```