Import dataset

Save as csv

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

kashti = sns.load_dataset('titanic')
kashti.head(5)
# dataset=pd.read_csv("./iris.csv")
```

Out[ ]:

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

Saving dataframe into csv

In [ ]:
```
# kashti.to_csv('kashti.csv')
dataset
```

Out[ ]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

In [ ]:
```
# basic statistics and summary
kashti.describe()
```

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 [ ]:

```
kashti
```

Out[ ]:

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

891 rows × 15 columns

In [ ]:

```
# drop some column and make new dataframe
kashti = kashti.drop(['deck', 'embark_town'], axis=1)
kashti
```

Out[ ]:

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

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male
886	0	2	male	27.0	0	0	13.0000	S	Second	man	True
887	1	1	female	19.0	0	0	30.0000	S	First	woman	False
888	0	3	female	NaN	1	2	23.4500	S	Third	woman	False
889	1	1	male	26.0	0	0	30.0000	C	First	man	True
890	0	3	male	32.0	0	0	7.7500	Q	Third	man	True

891 rows × 13 columns

In [ ]:
```
kashti.mean()
```

```
C:\Users\abdur\AppData\Local\Temp\ipykernel_20996\3332994036.py:1: FutureWarning: Dr
opping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is dep
recated; in a future version this will raise TypeError.  Select only valid columns b
efore calling the reduction.
  kashti.mean()
```

Out[ ]:
```
survived       0.383838
pclass         2.308642
age           29.699118
sibsp          0.523008
parch          0.381594
fare          32.204208
adult_male     0.602694
alone          0.602694
dtype: float64
```

In [ ]:
```
# total survived
kashti.value_counts('survived')
```

Out[ ]:
```
<bound method NDFrame._add_numeric_operations.<locals>.mean of survived
0    549
1    342
dtype: int64>
```

In [ ]:
```
# group femal and male
kashti.groupby(['sex']).mean()
```

Out[ ]:

sex	survived	pclass	age	sibsp	parch	fare	adult_male	alone
female	0.742038	2.159236	27.915709	0.694268	0.649682	44.479818	0.000000	0.401274
male	0.188908	2.389948	30.726645	0.429809	0.235702	25.523893	0.930676	0.712305

In [ ]:
```
# group by sex and their class
kashti.groupby(['sex', 'class']).mean()
```

Out[ ]:

sex	class	survived	pclass	age	sibsp	parch	fare	adult_male	alone
female	**First**	0.968085	1.0	34.611765	0.553191	0.457447	106.125798	0.000000	0.361702

sex	class	survived	pclass	age	sibsp	parch	fare	adult_male	alone
	Second	0.921053	2.0	28.722973	0.486842	0.605263	21.970121	0.000000	0.421053
	Third	0.500000	3.0	21.750000	0.895833	0.798611	16.118810	0.000000	0.416667
male	First	0.368852	1.0	41.281386	0.311475	0.278689	67.226127	0.975410	0.614754
	Second	0.157407	2.0	30.740707	0.342593	0.222222	19.741782	0.916667	0.666667
	Third	0.135447	3.0	26.507589	0.498559	0.224784	12.661633	0.919308	0.760807

In [ ]:
```
# group all children
kashti[kashti['age'] < 18].mean()
```

```
C:\Users\abdur\AppData\Local\Temp\ipykernel_20996\1148523881.py:1: FutureWarning: Dr
opping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is dep
recated; in a future version this will raise TypeError.  Select only valid columns b
efore calling the reduction.
  kashti[kashti['age'] < 18].mean()
```

Out[ ]:
```
survived       0.539823
pclass         2.584071
age            9.041327
sibsp          1.460177
parch          1.053097
fare          31.220798
adult_male     0.159292
alone          0.203540
dtype: float64
```

In [ ]:
```
kashti[kashti['age'] < 18].mean()
```

```
C:\Users\abdur\AppData\Local\Temp\ipykernel_20996\1148523881.py:1: FutureWarning: Dr
opping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is dep
recated; in a future version this will raise TypeError.  Select only valid columns b
efore calling the reduction.
  kashti[kashti['age'] < 18].mean()
```

Out[ ]:
```
survived       0.539823
pclass         2.584071
age            9.041327
sibsp          1.460177
parch          1.053097
fare          31.220798
adult_male     0.159292
alone          0.203540
dtype: float64
```

Find Unique value for multiple columns: ***Assignment***

In [ ]:
```
kashti.nunique()
```

Out[ ]:
```
survived         2
pclass           3
sex              2
age             88
sibsp            7
parch            7
fare           248
embarked         3
```

```
class            3
who              3
adult_male       2
deck             7
embark_town      3
alive            2
alone            2
dtype: int64
```

In [ ]:
```
kashti['fare'].nunique()
```

Out[ ]:
```
248
```

In [ ]:
```
kashti['age'].nunique()
```

Out[ ]:
```
88
```

In [ ]:
```
# Unique value through concatenation
pd.concat([ kashti['fare'], kashti['sex'], kashti['class']]).nunique()
```

Out[ ]:
```
253
```

In [ ]:
```
# unique value of multiple columns through np
np.unique(kashti[['fare', 'age']].values)
```

Out[ ]:
```
array([0.000000e+00, 4.200000e-01, 6.700000e-01, 7.500000e-01,
       8.300000e-01, 9.200000e-01, 1.000000e+00, 2.000000e+00,
       3.000000e+00, 4.000000e+00, 4.012500e+00, 5.000000e+00,
       6.000000e+00, 6.237500e+00, 6.437500e+00, 6.450000e+00,
       6.495800e+00, 6.750000e+00, 6.858300e+00, 6.950000e+00,
       6.975000e+00, 7.000000e+00, 7.045800e+00, 7.050000e+00,
       7.054200e+00, 7.125000e+00, 7.141700e+00, 7.225000e+00,
       7.229200e+00, 7.250000e+00, 7.312500e+00, 7.495800e+00,
       7.520800e+00, 7.550000e+00, 7.629200e+00, 7.650000e+00,
       7.725000e+00, 7.729200e+00, 7.733300e+00, 7.737500e+00,
       7.741700e+00, 7.750000e+00, 7.775000e+00, 7.787500e+00,
       7.795800e+00, 7.800000e+00, 7.829200e+00, 7.854200e+00,
       7.875000e+00, 7.879200e+00, 7.887500e+00, 7.895800e+00,
       7.925000e+00, 8.000000e+00, 8.029200e+00, 8.050000e+00,
       8.112500e+00, 8.137500e+00, 8.158300e+00, 8.300000e+00,
       8.362500e+00, 8.404200e+00, 8.433300e+00, 8.458300e+00,
       8.516700e+00, 8.654200e+00, 8.662500e+00, 8.683300e+00,
       8.712500e+00, 8.850000e+00, 9.000000e+00, 9.216700e+00,
       9.225000e+00, 9.350000e+00, 9.475000e+00, 9.483300e+00,
       9.500000e+00, 9.587500e+00, 9.825000e+00, 9.837500e+00,
       9.841700e+00, 9.845800e+00, 1.000000e+01, 1.017080e+01,
       1.046250e+01, 1.050000e+01, 1.051670e+01, 1.100000e+01,
       1.113330e+01, 1.124170e+01, 1.150000e+01, 1.200000e+01,
       1.227500e+01, 1.228750e+01, 1.235000e+01, 1.247500e+01,
       1.252500e+01, 1.265000e+01, 1.287500e+01, 1.300000e+01,
       1.341670e+01, 1.350000e+01, 1.379170e+01, 1.385830e+01,
       1.386250e+01, 1.400000e+01, 1.410830e+01, 1.440000e+01,
       1.445420e+01, 1.445830e+01, 1.450000e+01, 1.500000e+01,
       1.504580e+01, 1.505000e+01, 1.510000e+01, 1.524580e+01,
       1.550000e+01, 1.555000e+01, 1.574170e+01, 1.575000e+01,
       1.585000e+01, 1.590000e+01, 1.600000e+01, 1.610000e+01,
       1.670000e+01, 1.700000e+01, 1.740000e+01, 1.780000e+01,
       1.800000e+01, 1.875000e+01, 1.878750e+01, 1.900000e+01,
       1.925830e+01, 1.950000e+01, 1.996670e+01, 2.000000e+01,
```

```
       2.021250e+01, 2.025000e+01, 2.050000e+01, 2.052500e+01,
       2.057500e+01, 2.100000e+01, 2.107500e+01, 2.167920e+01,
       2.200000e+01, 2.202500e+01, 2.235830e+01, 2.252500e+01,
       2.300000e+01, 2.325000e+01, 2.345000e+01, 2.350000e+01,
       2.400000e+01, 2.415000e+01, 2.450000e+01, 2.500000e+01,
       2.546670e+01, 2.558750e+01, 2.592500e+01, 2.592920e+01,
       2.600000e+01, 2.625000e+01, 2.628330e+01, 2.628750e+01,
       2.638750e+01, 2.655000e+01, 2.700000e+01, 2.772080e+01,
       2.775000e+01, 2.790000e+01, 2.800000e+01, 2.850000e+01,
       2.871250e+01, 2.900000e+01, 2.912500e+01, 2.970000e+01,
       3.000000e+01, 3.007080e+01, 3.050000e+01, 3.069580e+01,
       3.100000e+01, 3.127500e+01, 3.138750e+01, 3.200000e+01,
       3.232080e+01, 3.250000e+01, 3.300000e+01, 3.350000e+01,
       3.400000e+01, 3.402080e+01, 3.437500e+01, 3.450000e+01,
       3.465420e+01, 3.500000e+01, 3.550000e+01, 3.600000e+01,
       3.650000e+01, 3.675000e+01, 3.700000e+01, 3.700420e+01,
       3.800000e+01, 3.850000e+01, 3.900000e+01, 3.940000e+01,
       3.960000e+01, 3.968750e+01, 4.000000e+01, 4.012500e+01,
       4.050000e+01, 4.100000e+01, 4.157920e+01, 4.200000e+01,
       4.240000e+01, 4.300000e+01, 4.400000e+01, 4.500000e+01,
       4.550000e+01, 4.600000e+01, 4.690000e+01, 4.700000e+01,
       4.710000e+01, 4.800000e+01, 4.900000e+01, 4.950000e+01,
       4.950420e+01, 5.000000e+01, 5.049580e+01, 5.100000e+01,
       5.147920e+01, 5.186250e+01, 5.200000e+01, 5.255420e+01,
       5.300000e+01, 5.310000e+01, 5.400000e+01, 5.500000e+01,
       5.544170e+01, 5.550000e+01, 5.590000e+01, 5.600000e+01,
       5.649580e+01, 5.692920e+01, 5.700000e+01, 5.797920e+01,
       5.800000e+01, 5.900000e+01, 5.940000e+01, 6.000000e+01,
       6.100000e+01, 6.117500e+01, 6.137920e+01, 6.197920e+01,
       6.200000e+01, 6.300000e+01, 6.335830e+01, 6.400000e+01,
       6.500000e+01, 6.600000e+01, 6.660000e+01, 6.930000e+01,
       6.955000e+01, 7.000000e+01, 7.050000e+01, 7.100000e+01,
       7.128330e+01, 7.350000e+01, 7.400000e+01, 7.525000e+01,
       7.629170e+01, 7.672920e+01, 7.728750e+01, 7.795830e+01,
       7.826670e+01, 7.885000e+01, 7.920000e+01, 7.965000e+01,
       8.000000e+01, 8.185830e+01, 8.217080e+01, 8.315830e+01,
       8.347500e+01, 8.650000e+01, 8.910420e+01, 9.000000e+01,
       9.107920e+01, 9.350000e+01, 1.064250e+02, 1.089000e+02,
       1.108833e+02, 1.132750e+02, 1.200000e+02, 1.336500e+02,
       1.345000e+02, 1.356333e+02, 1.465208e+02, 1.515500e+02,
       1.534625e+02, 1.648667e+02, 2.113375e+02, 2.115000e+02,
       2.217792e+02, 2.275250e+02, 2.475208e+02, 2.623750e+02,
       2.630000e+02, 5.123292e+02,          nan])
```

In [ ]: