Data Wrangling

In [1]:
```
#install libraries
#pip install pandas
#pip install seaborn
#pip install numpy
```

In [2]:
```
#import llibraries
import pandas as pd
import numpy as np
import seaborn as sns
```

In [3]:
```
kashti = sns.load_dataset('titanic')
ks1 = kashti
ks2 = kashti
ks = sns.load_dataset('titanic')
```

In [4]:
```
kashti.head()
```

Out[4]:

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

In [5]:
```
# simple operations (Math operator)
(kashti['age']+12).head(10)
```

Out[5]:
```
0    34.0
1    50.0
2    38.0
3    47.0
4    47.0
5     NaN
6    66.0
7    14.0
8    39.0
9    26.0
Name: age, dtype: float64
```

In [6]:
```
#where exactly missing values are?
kashti.isnull().sum()
```

Out[6]:
```
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 [7]:
```
#usd drop.na method
print(kashti.shape)
#kashti.dropna(subset=['deck'], axis=0, inplace=True) # this will remove specifically deck
#inplace = true modifies the data frame
```

```
(891, 15)
```

In [8]:
```
kashti.isnull().sum()
```

Out[8]:
```
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 [9]:
```
kashti = kashti.dropna()
kashti.isnull().sum() # remove na from whole dataframe
```

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           0
embark_town    0
alive          0
alone          0
dtype: int64
```

```
In [10]: kashti.shape

Out[10]: (182, 15)

In [11]: ks1.isnull().sum()

Out[11]: 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
```

Replacing missing values with the average of that column

```
In [12]: #finding an average (mean)
         mean = ks1['age'].mean()
         mean

Out[12]: 29.69911764705882

In [13]: # replacing nan with mean of the data (updating as well)
         ks1['age'] = ks1['age'].replace(np.nan, mean)

In [14]: ks1.isnull().sum()

Out[14]: 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 [15]:
```

```
ks1.dropna(subset=['deck'], axis=0, inplace=True)
ks1.dropna(subset=['embarked'], axis=0, inplace=True)
ks1.dropna(subset=['embark_town'], axis=0, inplace=True)
```

In [16]:

```
ks1.isnull().sum()
```

Out[16]:

```
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 Formatting

In [17]:

```
# know the data type and convert it into the known one
kashti.dtypes
```

Out[17]:

```
survived          int64
pclass            int64
sex              object
age             float64
sibsp             int64
parch             int64
fare            float64
embarked         object
class          category
who              object
adult_male         bool
deck           category
embark_town      object
alive            object
alone              bool
dtype: object
```

In [18]:

```
# use this method to convert datatype from one to another format
kashti['survived'] = kashti['survived'].astype("float64")
kashti.dtypes
```

```
C:\Users\eAgLe\AppData\Local\Temp/ipykernel_9780/2526506595.py:2: SettingWithCopyWarning:
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/inde
xing.html#returning-a-view-versus-a-copy
  kashti['survived'] = kashti['survived'].astype("float64")
```

Out[18]:

```
survived        float64
```

```
pclass             int64
sex               object
age              float64
sibsp              int64
parch              int64
fare             float64
embarked          object
class           category
who               object
adult_male          bool
deck            category
embark_town       object
alive             object
alone               bool
dtype: object
```

In [19]:
```
# here we will convert the age into days instead of years
ks1['age'] = ks1['age']*365
ks1.head(10)
```

Out[19]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	e
1	1	1	female	13870.000000	1	0	71.2833	C	First	woman	False	C	
3	1	1	female	12775.000000	1	0	53.1000	S	First	woman	False	C	S
6	0	1	male	19710.000000	0	0	51.8625	S	First	man	True	E	S
10	1	3	female	1460.000000	1	1	16.7000	S	Third	child	False	G	S
11	1	1	female	21170.000000	0	0	26.5500	S	First	woman	False	C	S
21	1	2	male	12410.000000	0	0	13.0000	S	Second	man	True	D	S
23	1	1	male	10220.000000	0	0	35.5000	S	First	man	True	A	S
27	0	1	male	6935.000000	3	2	263.0000	S	First	man	True	C	S
31	1	1	female	10840.177941	1	0	146.5208	C	First	woman	False	B	
52	1	1	female	17885.000000	1	0	76.7292	C	First	woman	False	D	

In [20]:
```
#here is the assignment to remove multipke 0's
```

In [21]:
```
# always rename afterwards
ks1.rename(columns={"age": "age in days"}, inplace=True)
ks1.head()
```

Out[21]:

	survived	pclass	sex	age in days	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_tc
1	1	1	female	13870.0	1	0	71.2833	C	First	woman	False	C	Cherbc
3	1	1	female	12775.0	1	0	53.1000	S	First	woman	False	C	Southamp
6	0	1	male	19710.0	0	0	51.8625	S	First	man	True	E	Southamp
10	1	3	female	1460.0	1	1	16.7000	S	Third	child	False	G	Southamp
11	1	1	female	21170.0	0	0	26.5500	S	First	woman	False	C	Southamp

In [22]:
```
ks1['age in days'] = ks1['age in days'].astype("int64")
ks1.head(10)
```

Out[22]:

	survived	pclass	sex	age in days	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_
1	1	1	female	13870	1	0	71.2833	C	First	woman	False	C	Cherk
3	1	1	female	12775	1	0	53.1000	S	First	woman	False	C	Southan
6	0	1	male	19710	0	0	51.8625	S	First	man	True	E	Southan
10	1	3	female	1460	1	1	16.7000	S	Third	child	False	G	Southan
11	1	1	female	21170	0	0	26.5500	S	First	woman	False	C	Southan
21	1	2	male	12410	0	0	13.0000	S	Second	man	True	D	Southan
23	1	1	male	10220	0	0	35.5000	S	First	man	True	A	Southan
27	0	1	male	6935	3	2	263.0000	S	First	man	True	C	Southan
31	1	1	female	10840	1	0	146.5208	C	First	woman	False	B	Cherk
52	1	1	female	17885	1	0	76.7292	C	First	woman	False	D	Cherk

Data Normalization

In [23]:
```
kashti.head()
```

Out[23]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town
1	1.0	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg
3	1.0	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton
6	0.0	1	male	54.0	0	0	51.8625	S	First	man	True	E	Southampton
10	1.0	3	female	4.0	1	1	16.7000	S	Third	child	False	G	Southampton
11	1.0	1	female	58.0	0	0	26.5500	S	First	woman	False	C	Southampton

In [24]:
```
ks4 = kashti[["age", "fare"]]
ks4.head()
```

Out[24]:

	age	fare
1	38.0	71.2833
3	35.0	53.1000
6	54.0	51.8625
10	4.0	16.7000

	age	fare
11	58.0	26.5500

Method of Normalization

In [25]:

```
# simle feature scaling
ks4['fare'] = ks4['fare']/ks4['fare'].max()
ks4['age'] = ks4['age']/ks4['age'].max()
ks4.head()
```

```
C:\Users\eAgLe\AppData\Local\Temp/ipykernel_9780/1199163970.py:2: SettingWithCopyWarning:
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/inde
xing.html#returning-a-view-versus-a-copy
  ks4['fare'] = ks4['fare']/ks4['fare'].max()
C:\Users\eAgLe\AppData\Local\Temp/ipykernel_9780/1199163970.py:3: SettingWithCopyWarning:
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/inde
xing.html#returning-a-view-versus-a-copy
  ks4['age'] = ks4['age']/ks4['age'].max()
```

Out[25]:

	age	fare
1	0.4750	0.139136
3	0.4375	0.103644
6	0.6750	0.101229
10	0.0500	0.032596
11	0.7250	0.051822

In [26]:

```
# Min - Max Method

ks4['fare'] = (ks4['fare']-ks4['fare'].min()) / (ks4['fare'].max()-ks4['fare'].min())
ks4.head()
```

```
C:\Users\eAgLe\AppData\Local\Temp/ipykernel_9780/410330791.py:3: SettingWithCopyWarning:
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/inde
xing.html#returning-a-view-versus-a-copy
  ks4['fare'] = (ks4['fare']-ks4['fare'].min()) / (ks4['fare'].max()-ks4['fare'].min())
```

Out[26]:

	age	fare
1	0.4750	0.139136
3	0.4375	0.103644
6	0.6750	0.101229
10	0.0500	0.032596

	age	fare
11	0.7250	0.051822

In [27]:
```python
# Z-score (standard score)
ks4['fare'] = (ks4['fare']-ks4['fare'].mean()) / ks4['fare'].std()
ks4.head()
```

```
C:\Users\eAgLe\AppData\Local\Temp/ipykernel_9780/1430778810.py:2: SettingWithCopyWarning:
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/inde
xing.html#returning-a-view-versus-a-copy
  ks4['fare'] = (ks4['fare']-ks4['fare'].mean()) / ks4['fare'].std()
```

Out[27]:

	age	fare
1	0.4750	-0.099835
3	0.4375	-0.337554
6	0.6750	-0.353732
10	0.0500	-0.813428
11	0.7250	-0.684654

In [28]:
```python
# log transformation
ks['fare'] = np.log(ks['fare'])
ks.head()
```

```
H:\download\Anaconda\lib\site-packages\pandas\core\arraylike.py:364: RuntimeWarning: divide by zer
o encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

Out[28]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town
0	0	3	male	22.0	1	0	1.981001	S	Third	man	True	NaN	Southampton
1	1	1	female	38.0	1	0	4.266662	C	First	woman	False	C	Cherbourg
2	1	3	female	26.0	0	0	2.070022	S	Third	woman	False	NaN	Southampton
3	1	1	female	35.0	1	0	3.972177	S	First	woman	False	C	Southampton
4	0	3	male	35.0	0	0	2.085672	S	Third	man	True	NaN	Southampton

Binning

In [29]:
```python
kashti = sns.load_dataset('titanic')
```

assignment given in the video

In [30]:
```python
bins = np.linspace(min(kashti['age']), max(kashti['age']), 4)
```

```
age_groups = ["Bachay", "Jawan", "Boorhay"]
kashti['age'] = pd.cut(kashti['age'], bins, labels=age_groups, include_lowest=True)
kashti['age']
# how this will change the anames in dataset based on grouping? (Assignment)
```

Out[30]:

```
0       Bachay
1        Jawan
2       Bachay
3        Jawan
4        Jawan
         ...
886      Jawan
887     Bachay
888        NaN
889     Bachay
890      Jawan
Name: age, Length: 891, dtype: category
Categories (3, object): ['Bachay' < 'Jawan' < 'Boorhay']
```

In [31]:

```
kashti.head(10)
```

Out[31]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_t
0	0	3	male	Bachay	1	0	7.2500	S	Third	man	True	NaN	Southam
1	1	1	female	Jawan	1	0	71.2833	C	First	woman	False	C	Cherb
2	1	3	female	Bachay	0	0	7.9250	S	Third	woman	False	NaN	Southam
3	1	1	female	Jawan	1	0	53.1000	S	First	woman	False	C	Southam
4	0	3	male	Jawan	0	0	8.0500	S	Third	man	True	NaN	Southam
5	0	3	male	NaN	0	0	8.4583	Q	Third	man	True	NaN	Queens
6	0	1	male	Boorhay	0	0	51.8625	S	First	man	True	E	Southam
7	0	3	male	Bachay	3	1	21.0750	S	Third	child	False	NaN	Southam
8	1	3	female	Jawan	0	2	11.1333	S	Third	woman	False	NaN	Southam
9	1	2	female	Bachay	1	0	30.0708	C	Second	child	False	NaN	Cherb

converting categories into dummies

- easy to use for computation
- Male Female (0,1)

how to use get dummies to change data inside a dataframe (Assignment)

1st method

In [51]:

```
pd.get_dummies(ks["sex"],prefix="Sex",columns=ks["sex"])
```

Out[51]:

	Sex_female	Sex_male

	Sex_female	Sex_male
0	0	1
1	1	0
2	1	0
3	1	0
4	0	1
...	...	...
886	0	1
887	1	0
888	1	0
889	0	1
890	0	1

891 rows × 2 columns

In [56]:
```
# df = load_data() # reset dataframe
df = pd.get_dummies(ks, columns=['sex'])
df
```

Out[56]:

	survived	pclass	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	a
0	0	3	22.0	1	0	1.981001	S	Third	man	True	NaN	Southampton	
1	1	1	38.0	1	0	4.266662	C	First	woman	False	C	Cherbourg	
2	1	3	26.0	0	0	2.070022	S	Third	woman	False	NaN	Southampton	
3	1	1	35.0	1	0	3.972177	S	First	woman	False	C	Southampton	
4	0	3	35.0	0	0	2.085672	S	Third	man	True	NaN	Southampton	
...	...	...	...	...	...	...	...	...	...	...	...	...	
886	0	2	27.0	0	0	2.564949	S	Second	man	True	NaN	Southampton	
887	1	1	19.0	0	0	3.401197	S	First	woman	False	B	Southampton	
888	0	3	NaN	1	2	3.154870	S	Third	woman	False	NaN	Southampton	
889	1	1	26.0	0	0	3.401197	C	First	man	True	C	Cherbourg	
890	0	3	32.0	0	0	2.047693	Q	Third	man	True	NaN	Queenstown	

891 rows × 16 columns

2nd method

In [58]:
```
ks = sns.load_dataset('titanic')
ks.head()
```

Out[58]:

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

In [60]:

```
#here i can directly call the function and pass the value
ks['sex'] = ks['sex'].map({'female': 1, 'male': 0})
ks.head()
```

Out[60]:

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

In [ ]: