import libraries

In [ ]:

```
import numpy as np
import pandas as pd
import seaborn as sns
```

In [ ]:

```
kashti= sns.load_dataset("titanic")
Ks1= kashti
Ks2= kashti
Ks= sns.load_dataset("titanic")
```

In [ ]:

```
kashti.head()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	e
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	S
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	S
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	S
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	S

In [ ]:

```
# simple operations (Mathematics operator)
(kashti["age"]+1).head(10) # "1 is added in previous age"
```

Out[ ]:

```
0    23.0
1    39.0
2    27.0
3    36.0
4    36.0
5     NaN
6    55.0
7     3.0
8    28.0
9    15.0
Name: age, dtype: float64
```

Steps for Data wrangling

Step 1- dealing with missing values

1- In a data set missing values are either '?' or N/A or NaN (not a number) or '0' or a blank cell.

Steps:\ Recheck twice if any mistake is there.\ Recollect the data\ Remove the variable (column or row) having missing value, if doesn't matter.\ Replace the missing value.\

```
1. **How?**\
    1. Average value of entire variable or similar data point.\
    2. Frequency or MODE replacement.\
        3. Replaced based on other function (data sampller knows that)\
        4. ML algorith can be used.\
        5. Leave it as it is.\
    2. **Why?**\
        1. Its better because not data is lost.\
        2. Less accurate
```

In [ ]:

```
# Where is the missing (isnull) values are?
kashti.isnull().sum() #to count no.of missing values in a variable
```

Out[ ]:

```
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 [ ]:

```
# use drop.na method
print (kashti.shape)
kashti.dropna(subset= ['deck'], axis=0, inplace=True) # this will remove specifically
#inplace= True modifies the data frame, "false" wouldn't.
```

```
(891, 15)
```

In [ ]:

```
# To remove, Recheck and find the number of missing value
#use drop.na method
kashti.dropna()

# to update the main data frame
kashti= kashti.dropna().isnull().sum() # remove na from whole data
```

In [ ]:

```
kashti.shape
```

Out[ ]:

```
(15,)
```

In [ ]:

```
Ks1.isnull().sum()
```

Out[ ]:

```
survived        0
pclass          0
sex             0
age            19
sibsp           0
parch           0
fare            0
```

```
embarked       2
class          0
who            0
adult_male     0
deck           0
embark_town    2
alive          0
alone          0
dtype: int64
```

Step-2 Replace the missing values with average of that column

In [ ]:

```
# finding an average
mean= Ks1['age'].mean()
mean
```

Out[ ]:

```
35.77945652173913
```

In [ ]:

```
# Replace NaN with the mean of data (updating as well)
Ks1['age']= Ks1['age'].replace(np.nan, mean) # nan has been used as a numpy array
```

In [ ]:

```
Ks1.isnull().sum()
```

Out[ ]:

```
survived      0
pclass        0
sex           0
age           0
sibsp         0
parch         0
fare          0
embarked      2
class         0
who           0
adult_male    0
deck          0
embark_town   2
alive         0
alone         0
dtype: int64
```

Data Formatting

```
    - To standardize the data
    - Ensures that data is consistent and understandable
      - Easy to gather
      - Easy to workwith
         1.Chakwal(CKL) # not write a misture, Either 'Chakwal' , or 'CKL'
    ,Its called Data standardization or formatting.
         2. Islamabad (ISB)
         3. Lahore (LHR)
         4. Conver g to kg or similar unit for all.
         5. Standard unit in each column
         6. e.g= ft != cm
```

In [ ]:

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

```
Ks1.dtypes
```

Out[ ]:

```
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 [ ]:

```
# use thiis method to convert data type from one to another format
Ks1['survived'] = Ks1['survived'].astype("float")
Ks1.dtypes
```

Out[ ]:

```
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 [ ]:

```
Ks1['survived'] = Ks1['survived'].astype("int64")
Ks1.dtypes
```

Out[ ]:

```
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 [ ]:

```
Ks2['age'] = Ks2['age'].astype("int64")
Ks2.dtypes
```

Out[ ]:

```
survived            int64
pclass              int64
sex                object
age                 int64
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 [ ]:

```
Ks2['fare'] = Ks2['fare'].astype("int64")
Ks2.dtypes
```

Out[ ]:

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

In [ ]:

```
# here we will convert age into days, rather than years
Ks2['age in days']= Ks2['age in days']*365
Ks2.head(10) #data for 10 days
```

Out[ ]:

	survived	pclass	sex	age in days	sibsp	parch	fare	embarked	class	who	adult_male	deck
1	1	1	female	13870	1	0	71	C	First	woman	False	C
3	1	1	female	12775	1	0	53	S	First	woman	False	C
6	0	1	male	19710	0	0	51	S	First	man	True	E
10	1	3	female	1460	1	1	16	S	Third	child	False	G

	survived	pclass	sex	age in days	sibsp	parch	fare	embarked	class	who	adult_male	deck
11	1	1	female	21170	0	0	26	S	First	woman	False	C
21	1	2	male	12410	0	0	13	S	Second	man	True	D
23	1	1	male	10220	0	0	35	S	First	man	True	A
27	0	1	male	6935	3	2	263	S	First	man	True	C
31	1	1	female	12775	1	0	146	C	First	woman	False	B
52	1	1	female	17885	1	0	76	C	First	woman	False	D

In [ ]:

```
# always rename afterwards
Ks2.rename(columns= {"age": "age in days"}, inplace= True)
Ks2.head()
```

Out[ ]:

	survived	pclass	sex	age in days	sibsp	parch	fare	embarked	class	who	adult_male	deck	e
1	1	1	female	13870	1	0	71	C	First	woman	False	C	
3	1	1	female	12775	1	0	53	S	First	woman	False	C	S
6	0	1	male	19710	0	0	51	S	First	man	True	E	S
10	1	3	female	1460	1	1	16	S	Third	child	False	G	S
11	1	1	female	21170	0	0	26	S	First	woman	False	C	S

Data Normalization

- Uniform the Data
- They have same impact
- Also for computational reasons
- Whole data set should be within '0' to '1'. So it could be plotted.

In [ ]:

```
kashti.head()
```

Out[ ]:

```
survived    0
pclass      0
sex         0
age         0
sibsp       0
dtype: int64
```

In [ ]:

```
Ks4 = kashti[["age", "fare"]]
Ks4.head()
```

Out[ ]:

```
age     0
fare    0
```

```
dtype: int64
```

Methods of Normalization

1. Simple feature scalling x(new) = x(old)/ x(max)
2. Min-Max method
3. Z-score (standard score) -3 to +3
4. Log transformation

In [ ]:

```
# simple feature scalling
Ks4['fare']= Ks4['fare']/Ks4['fare'].max()
Ks4.head()
```

```
C:\Users\Javeria\AppData\Local\Temp\ipykernel_13752\477595571.py:2: RuntimeWarning: inva
lid value encountered in longlong_scalars
  Ks4['fare']= Ks4['fare']/Ks4['fare'].max()
```

Out[ ]:

```
age     0.0
fare    NaN
dtype: float64
```

In [ ]:

```
# min-max method
Ks4['fare']= (Ks4['fare']-Ks4['fare'].min()) / Ks4['fare'].max()- Ks4['fare'].min()
Ks4.head()
```

Out[ ]:

```
age     0.0
fare    NaN
dtype: float64
```

In [ ]:

```
# Z-score method
Ks4['fare'] = (Ks4['fare']-Ks4['fare'].mean()) / Ks4['fare'].std()
Ks4.head()
```

Out[ ]:

```
age     0.0
fare    NaN
dtype: float64
```

In [ ]:

```
# Log transformation
Ks['fare'] = np.log(Ks['fare'])
Ks.head()
```

```
C:\Users\Javeria\AppData\Local\Programs\Python\Python310\lib\site-packages\pandas\core\a
rraylike.py:364: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
```

Out[ ]:

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

Binning

- Grouping values into smaller number of values (bins)
- Convert numeric into categories (Child, adult, old)
- To have better understanding of groups\ -low Vs mid Vs high price

In [ ]:

```
bins = np.linspace(min(kashti['age']), max(kashti['age']), 15000)
age_groups= ["Child, adult, old"]
kashti['age']= pd.cut(kashti['age'], bins, labels= age_groups, include_lowest= True)
kashti['age']
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
c:\Users\Javeria\Desktop\Machine learning\Data_wranglilng.ipynb Cell 38' in <module>
----> <a href='vscode-notebook-cell:/c%3A/Users/Javeria/Desktop/Machine%20learning/Data_wranglilng.ipynb#ch0000037?line=0'>1</a> bins = np.linspace(min(kashti['age']), max(kashti['age']), 1500.3)
      <a href='vscode-notebook-cell:/c%3A/Users/Javeria/Desktop/Machine%20learning/Data_wranglilng.ipynb#ch0000037?line=1'>2</a> age_groups= ["Child, adult, old"]
      <a href='vscode-notebook-cell:/c%3A/Users/Javeria/Desktop/Machine%20learning/Data_wranglilng.ipynb#ch0000037?line=2'>3</a> kashti['age']= pd.cut(kashti['age'], bins, labels= age_groups, include_lowest= True)

TypeError: 'numpy.int64' object is not iterable
```

Converting categories into dummies

- easy to use for computation
- male, female (0, 1)

In [ ]:

```
pd.get_dummies(Ks1['sex'])
```

Out[ ]:

	female	male
1	1	0
3	1	0
6	0	1
10	1	0
11	1	0
...	...	...
871	1	0
872	0	1
879	1	0
887	1	0
889	0	1

203 rows × 2 columns

Assignments

1. Transfer into dummy values
2. pd.get_dummies(Ks1['sex']) how to use get dummies to change data inside a data frame?
3. How binning will change the name in the dataset based on grouping?