Data wrangling assignment

1. Handling missing values
2. Remove zero's after decimal
3. Categorize column values into three groups
4. Replace column values with dummies

In [ ]:
```
# Import libraries
import numpy as np
import pandas as pd
import seaborn as sns
```

In [ ]:
```
# Load data
kashti = sns.load_dataset('titanic')
kashti.head()
```

Out[ ]:

	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

1. Handling missing values

In [ ]:
```
# Check missing values
kashti.isnull().sum()
```

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
```

Column `deck` has high ratio of missing values if it is more than **75%** then we will have to drop the column because it won't be helpful for data analysis.

In [ ]:
```
# Calculate percentage of missing values in "deck"
missing_deck_percentage = kashti['deck'].isnull().sum()*100/len(kashti['deck'])
print(f'Percentage of missing value in deck: {missing_deck_percentage:.2f}%')
```

```
Percentage of missing value in deck: 77.22%
```

In [ ]:
```
# Drop column 'deck'
kashti.drop(columns=['deck'], inplace=True)
```

Drop rows of `age`, `embarked` and `embark_town` where the values are missing.

In [ ]:
```
# Drop missing values in 'embarked' and 'embark_town'
kashti.dropna(subset=['age', 'embarked', 'embark_town'], inplace=True)
```

In [ ]:
```
# Check the missing values again
kashti.isnull().sum()
```

Out[ ]:
```
survived       0
pclass         0
sex            0
age            0
sibsp          0
parch          0
fare           0
embarked       0
class          0
who            0
adult_male     0
embark_town    0
alive          0
alone          0
dtype: int64
```

2. Remove zero's after decimal point

Remove the zero's after decimal point in `age` columns values.

In [ ]:
```
# Get columns information
kashti.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 712 entries, 0 to 890
Data columns (total 14 columns):
 #   Column       Non-Null Count  Dtype   
---  ------       --------------  -----   
 0   survived     712 non-null    int64   
 1   pclass       712 non-null    int64   
 2   sex          712 non-null    object  
 3   age          712 non-null    float64 
 4   sibsp        712 non-null    int64   
 5   parch        712 non-null    int64   
 6   fare         712 non-null    float64 
 7   embarked     712 non-null    object  
 8   class        712 non-null    category
 9   who          712 non-null    object  
 10  adult_male   712 non-null    bool    
 11  embark_town  712 non-null    object  
 12  alive        712 non-null    object  
 13  alone        712 non-null    bool    
dtypes: bool(2), category(1), float64(2), int64(4), object(5)
memory usage: 69.0+ KB
```

In [ ]:
```
# Convert 'age' values from float to int to remove decimal points
kashti['age'] = kashti['age'].astype('int64')
kashti.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 712 entries, 0 to 890
Data columns (total 14 columns):
 #   Column       Non-Null Count  Dtype   
---  ------       --------------  -----   
 0   survived     712 non-null    int64   
 1   pclass       712 non-null    int64   
 2   sex          712 non-null    object  
 3   age          712 non-null    int64   
 4   sibsp        712 non-null    int64   
 5   parch        712 non-null    int64   
 6   fare         712 non-null    float64 
 7   embarked     712 non-null    object  
 8   class        712 non-null    category
 9   who          712 non-null    object  
 10  adult_male   712 non-null    bool    
 11  embark_town  712 non-null    object  
 12  alive        712 non-null    object  
 13  alone        712 non-null    bool    
dtypes: bool(2), category(1), float64(1), int64(5), object(5)
memory usage: 69.0+ KB
```

In [ ]:
```
# View results
kashti.head()
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	embark_town	alive	alone
0	0	3	male	22	1	0	7.2500	S	Third	man	True	Southampton	no	False
1	1	1	female	38	1	0	71.2833	C	First	woman	False	Cherbourg	yes	False
2	1	3	female	26	0	0	7.9250	S	Third	woman	False	Southampton	yes	True
3	1	1	female	35	1	0	53.1000	S	First	woman	False	Southampton	yes	False
4	0	3	male	35	0	0	8.0500	S	Third	man	True	Southampton	no	True

3. Categorize column values into three groups

Make 3 categories of `bachay, Jawan, and Boorhay` in column `age`.

In [ ]:
```
# Define range to ages to categorize
bins = [0, 20, 40, 100]
# Make age groups
age_groups = ['Bachay', 'Jawan', 'Boorhay']
# Make new column 'age groups' of three categories
kashti['age groups'] = pd.cut(kashti['age'], bins, labels=age_groups, include_lowest=True)

kashti.head(10)
```

Out[ ]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	embark_town	alive	alone	age groups
0	0	3	male	22	1	0	7.2500	S	Third	man	True	Southampton	no	False	Jawan
1	1	1	female	38	1	0	71.2833	C	First	woman	False	Cherbourg	yes	False	Jawan
2	1	3	female	26	0	0	7.9250	S	Third	woman	False	Southampton	yes	True	Jawan
3	1	1	female	35	1	0	53.1000	S	First	woman	False	Southampton	yes	False	Jawan
4	0	3	male	35	0	0	8.0500	S	Third	man	True	Southampton	no	True	Jawan
6	0	1	male	54	0	0	51.8625	S	First	man	True	Southampton	no	True	Boorhay
7	0	3	male	2	3	1	21.0750	S	Third	child	False	Southampton	no	False	Bachay
8	1	3	female	27	0	2	11.1333	S	Third	woman	False	Southampton	yes	False	Jawan
9	1	2	female	14	1	0	30.0708	C	Second	child	False	Cherbourg	yes	False	Bachay
10	1	3	female	4	1	1	16.7000	S	Third	child	False	Southampton	yes	False	Bachay

4. Replace columns with dummies

Replace values of `sex` with dummies.

In [ ]:
```
# Get one hot encoding of column 'sex'
one_hot = pd.get_dummies(kashti['sex'])
# Join the encoded dataframe (one_hot)
kashti = kashti.join(one_hot)
# Drop 'sex' column
kashti = kashti.drop(columns=['sex'], axis=1)

kashti.head()
```

Out[ ]:

	survived	pclass	age	sibsp	parch	fare	embarked	class	who	adult_male	embark_town	alive	alone	age groups	female	male
0	0	3	22	1	0	7.2500	S	Third	man	True	Southampton	no	False	Jawan	0	1
1	1	1	38	1	0	71.2833	C	First	woman	False	Cherbourg	yes	False	Jawan	1	0
2	1	3	26	0	0	7.9250	S	Third	woman	False	Southampton	yes	True	Jawan	1	0
3	1	1	35	1	0	53.1000	S	First	woman	False	Southampton	yes	False	Jawan	1	0
4	0	3	35	0	0	8.0500	S	Third	man	True	Southampton	no	True	Jawan	0	1