Muhammad Arham Adeel

Pandas 2nd Assignment

06- Reverse Row Order

In [1]:
```
import seaborn as sns
import pandas as pd

df = sns.load_dataset("titanic")
df.head()
```

Out[1]:

	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 [2]:
```
df.loc[::-1]
```

Out[2]:

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

891 rows × 15 columns

In [3]:

```
# here i can see only 5 instances

df.loc[::-1].head()
```

Out[3]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck
890	0	3	male	32.0	0	0	7.75	Q	Third	man	True	NaN
889	1	1	male	26.0	0	0	30.00	C	First	man	True	C
888	0	3	female	NaN	1	2	23.45	S	Third	woman	False	NaN
887	1	1	female	19.0	0	0	30.00	S	First	woman	False	B
886	0	2	male	27.0	0	0	13.00	S	Second	man	True	NaN

In [4]:

```
df.loc[::-1].reset_index(drop=True).head()
```

Out[4]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	e
0	0	3	male	32.0	0	0	7.75	Q	Third	man	True	NaN	
1	1	1	male	26.0	0	0	30.00	C	First	man	True	C	
2	0	3	female	NaN	1	2	23.45	S	Third	woman	False	NaN	S
3	1	1	female	19.0	0	0	30.00	S	First	woman	False	B	S
4	0	2	male	27.0	0	0	13.00	S	Second	man	True	NaN	S

07- Reverse Column order

In [5]:

```
df.loc[:,::-1].head()
```

Out[5]:

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

08- Select a column by dtype

In [6]:

```
df.dtypes
```

Out[6]:

```
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 [9]:
```
# only select those who have numeric types

df.select_dtypes(include = ["number"]).head()
```

Out[9]:

	survived	pclass	age	sibsp	parch	fare
0	0	3	22.0	1	0	7.2500
1	1	1	38.0	1	0	71.2833
2	1	3	26.0	0	0	7.9250
3	1	1	35.0	1	0	53.1000
4	0	3	35.0	0	0	8.0500

In [10]:
```
# only select those who have object types

df.select_dtypes(include = ["object"]).head()
```

Out[10]:

	sex	embarked	who	embark_town	alive
0	male	S	man	Southampton	no
1	female	C	woman	Cherbourg	yes
2	female	S	woman	Southampton	yes
3	female	S	woman	Southampton	yes
4	male	S	man	Southampton	no

In [11]:
```
# only select those who have categpry types

df.select_dtypes(include = ["category"]).head()
```

Out[11]:

	class	deck
0	Third	NaN
1	First	C

	class	deck
2	Third	NaN
3	First	C
4	Third	NaN

In [12]:
```
# only select those who have numeric,object and category types

df.select_dtypes(include = ["number","object","category"]).head()
```

Out[12]:

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

In [14]:
```
# select only those where variable type is not object

df.select_dtypes(exclude = ["object"]).head()
```

Out[14]:

	survived	pclass	age	sibsp	parch	fare	class	adult_male	deck	alone
0	0	3	22.0	1	0	7.2500	Third	True	NaN	False
1	1	1	38.0	1	0	71.2833	First	False	C	False
2	1	3	26.0	0	0	7.9250	Third	False	NaN	True
3	1	1	35.0	1	0	53.1000	First	False	C	False
4	0	3	35.0	0	0	8.0500	Third	True	NaN	True

In [15]:
```
# select only those where variable type is not numeric

df.select_dtypes(exclude = ["number"]).head()
```

Out[15]:

	sex	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	male	S	Third	man	True	NaN	Southampton	no	False
1	female	C	First	woman	False	C	Cherbourg	yes	False
2	female	S	Third	woman	False	NaN	Southampton	yes	True
3	female	S	First	woman	False	C	Southampton	yes	False
4	male	S	Third	man	True	NaN	Southampton	no	True

09- Convert strings to numbers

In [16]:
```
df = pd.DataFrame({
    'col_A': ["1.1","2","3","4","5","6"],
    'col_B': ["7","8","9","10","11","12"]
})

df
```

Out[16]:

	col_A	col_B
0	1.1	7
1	2	8
2	3	9
3	4	10
4	5	11
5	6	12

In [17]:
```
df.dtypes
```

Out[17]:
```
col_A    object
col_B    object
dtype: object
```

In [21]:
```
df.astype({"col_A": "float", "col_B": "int64"}).dtypes
```

Out[21]:
```
col_A    float64
col_B      int64
dtype: object
```

Methods of to numeric(eroor)

1. ignore, if errors = "ignore", then invalid parsing will return the input.
2. coerce, if errors = "coerce", then non-numeric will be set as NaN.
3. raise (default), if errors = "raise", then invalid parsing will raise an error

In [23]:
```
pd.to_numeric(df["col_A"], errors="coerce")
```

Out[23]:
```
0    1.1
1    2.0
2    3.0
3    4.0
4    5.0
5    6.0
Name: col_A, dtype: float64
```

10- Reduce DataFrame size

In [24]:
```
df = sns.load_dataset("titanic")
df.shape
```

Out[24]:
```
(891, 15)
```

In [25]:
```
# Returns the memory usage of each column in bytes.
df.memory_usage(deep=True)
```

Out[25]:
```
Index           128
survived       7128
pclass         7128
sex           54979
age            7128
sibsp          7128
parch          7128
fare           7128
embarked      51626
class          1186
who           54168
adult_male      891
deck           1597
embark_town   60103
alive         52911
alone           891
dtype: int64
```

In [26]:
```
df.sample(frac=0.1).shape
```

Out[26]:
```
(89, 15)
```

In [ ]: