Introduction to Pandas

Loading the dependencies

In [1]:

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

Loading the dataset

In [2]:

```
data = pd.read_csv('./titanic.csv')
data.head()
```

Out[2]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

droping the columns

In [3]:

```
data.drop(columns=['PassengerId', 'Cabin', 'Ticket', 'Name'], inplace=True)
```

In [4]:

```
data.head(10)
```

Out[4]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.2500	S
1	1	1	female	38.0	1	0	71.2833	C
2	1	3	female	26.0	0	0	7.9250	S
3	1	1	female	35.0	1	0	53.1000	S
4	0	3	male	35.0	0	0	8.0500	S
5	0	3	male	NaN	0	0	8.4583	Q
6	0	1	male	54.0	0	0	51.8625	S
7	0	3	male	2.0	3	1	21.0750	S
8	1	3	female	27.0	0	2	11.1333	S
9	1	2	female	14.0	1	0	30.0708	C

groupby

Gender wise survival

In [5]:

```
Sex_Survived_Sum = data.groupby(['Sex'])['Survived'].sum()
```

In [6]:

```
Sex_Survived_Sum
```

Out[6]:

```
Sex
female    233
male      109
Name: Survived, dtype: int64
```

In [7]:

```
Sex_Survived_Average = data.groupby(['Sex'])['Survived'].mean()*100
```

In [8]:

```
Sex_Survived_Average
```

Out[8]:

```
Sex
female    74.203822
male      18.890815
Name: Survived, dtype: float64
```

Querying the data

In [9]:

```
Sex_Age_20 = data.query("Sex == 'male' & Age == 22.0")
```

In [10]:

```
Sex_Age_20
```

60	0	3	male	22.0	0	0	7.2292	C
80	0	3	male	22.0	0	0	9.0000	S
112	0	3	male	22.0	0	0	8.0500	S
212	0	3	male	22.0	0	0	7.2500	S
225	0	3	male	22.0	0	0	9.3500	S
243	0	3	male	22.0	0	0	7.1250	S
287	0	3	male	22.0	0	0	7.8958	S
320	0	3	male	22.0	0	0	7.2500	S
373	0	1	male	22.0	0	0	135.6333	C
395	0	3	male	22.0	0	0	7.7958	S
478	0	3	male	22.0	0	0	7.5208	S
521	0	3	male	22.0	0	0	7.8958	S
553	1	3	male	22.0	0	0	7.2250	C

In [11]:

```
Female_Age_30 = data.query("Sex == 'female' & Age == 30.0")
Female_Age_30
```

Out[11]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
79	1	3	female	30.0	0	0	12.4750	S
257	1	1	female	30.0	0	0	86.5000	S
309	1	1	female	30.0	0	0	56.9292	C
322	1	2	female	30.0	0	0	12.3500	Q
520	1	1	female	30.0	0	0	93.5000	S
534	0	3	female	30.0	0	0	8.6625	S
537	1	1	female	30.0	0	0	106.4250	C
726	1	2	female	30.0	3	0	21.0000	S
747	1	2	female	30.0	0	0	13.0000	S
799	0	3	female	30.0	1	1	24.1500	S
842	1	1	female	30.0	0	0	31.0000	C

In [12]:

```
Sex_Age_20_SibSp_1 = data.query("Sex == 'male' & Age == 22.0 & SibSp == 1")
Sex_Age_20_SibSp_1
```

Out[12]:

	Survived	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	1	0	7.25	S

In [13]:

```
pivot_01 = data.pivot_table(values='Fare', index='Sex', columns='Pclass', aggfunc='sum')
```

In [14]:

```
pivot_01
```

Out[14]:

Pclass	**1**	**2**	**3**
Sex			
female	9975.8250	1669.7292	2321.1086
male	8201.5875	2132.1125	4393.5865

In [15]:

```
pivot_02 = data.pivot_table(values='Fare', index='Embarked', columns='Pclass', aggfunc='s
```

In [16]:

```
pivot_02
```

Out[16]:

Pclass	1	2	3
Embarked			
C	8901.0750	431.0917	740.1295
Q	180.0000	37.0500	805.2043
S	8936.3375	3333.7000	5169.3613

In [17]:

```
data.select_dtypes('int64')
```

Out[17]:

	Survived	Pclass	SibSp	Parch
0	0	3	1	0
1	1	1	1	0
2	1	3	0	0
3	1	1	1	0
4	0	3	0	0
...	...	...	...	...
886	0	2	0	0
887	1	1	0	0
888	0	3	1	2
889	1	1	0	0
890	0	3	0	0

891 rows × 4 columns

insert

This will insert a new column in the desired place. For this purpose let's create a new column first using np.random

In [18]:

```
column = np.random.randint(0,100, size = len(data))
```

In [19]:

```
data.insert(4, 'column', column)
```

In [20]:

```
data.head()
```

Out[20]:

	Survived	Pclass	Sex	Age	column	SibSp	Parch	Fare	Embarked
0	0	3	male	22.0	77	1	0	7.2500	S
1	1	1	female	38.0	4	1	0	71.2833	C
2	1	3	female	26.0	16	0	0	7.9250	S
3	1	1	female	35.0	54	1	0	53.1000	S
4	0	3	male	35.0	3	0	0	8.0500	S

cumsum()

In [21]:

```
data[['Fare', 'Age']].cumsum()
```

Out[21]:

	Fare	Age
0	7.2500	22.00
1	78.5333	60.00
2	86.4583	86.00
3	139.5583	121.00
4	147.6083	156.00
...	...	...
886	28602.7493	21128.17
887	28632.7493	21147.17
888	28656.1993	NaN
889	28686.1993	21173.17
890	28693.9493	21205.17

891 rows × 2 columns

where

In [22]:

```
data.where(data.column>50)
```

Out[22]:

	Survived	Pclass	Sex	Age	column	SibSp	Parch	Fare	Embarked
0	0.0	3.0	male	22.0	77.0	1.0	0.0	7.25	S
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	1.0	1.0	female	35.0	54.0	1.0	0.0	53.10	S
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...
886	0.0	2.0	male	27.0	98.0	0.0	0.0	13.00	S
887	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
888	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
889	1.0	1.0	male	26.0	87.0	0.0	0.0	30.00	C
890	0.0	3.0	male	32.0	62.0	0.0	0.0	7.75	Q

891 rows × 9 columns

In [23]:

```
data.Pclass.unique()
```

Out[23]:

```
array([3, 1, 2], dtype=int64)
```

In [24]:

```
data.Pclass.nunique()
```

Out[24]:

```
3
```

In [25]:

```
data['Rank'] = data.Fare.rank()
```

In [26]:

```
data
```

Out[26]:

	Survived	Pclass	Sex	Age	column	SibSp	Parch	Fare	Embarked	Rank
0	0	3	male	22.0	77	1	0	7.2500	S	77.0
1	1	1	female	38.0	4	1	0	71.2833	C	789.0
2	1	3	female	26.0	16	0	0	7.9250	S	232.5
3	1	1	female	35.0	54	1	0	53.1000	S	748.0
4	0	3	male	35.0	3	0	0	8.0500	S	264.0
...	...	...	...	...	...	...	...	...	...	...
886	0	2	male	27.0	98	0	0	13.0000	S	407.5
887	1	1	female	19.0	17	0	0	30.0000	S	654.5
888	0	3	female	NaN	16	1	2	23.4500	S	546.5
889	1	1	male	26.0	87	0	0	30.0000	C	654.5
890	0	3	male	32.0	62	0	0	7.7500	Q	123.5

891 rows × 10 columns

In [27]:

```
data.sort_values(by='Rank')
```

Out[27]:

	Survived	Pclass	Sex	Age	column	SibSp	Parch	Fare	Embarked	Rank
271	1	3	male	25.0	79	0	0	0.0000	S	8.0
597	0	3	male	49.0	50	0	0	0.0000	S	8.0
302	0	3	male	19.0	35	0	0	0.0000	S	8.0
633	0	1	male	NaN	74	0	0	0.0000	S	8.0
277	0	2	male	NaN	5	0	0	0.0000	S	8.0
...	...	...	...	...	...	...	...	...	...	...
438	0	1	male	64.0	68	1	4	263.0000	S	886.5
341	1	1	female	24.0	67	3	2	263.0000	S	886.5
737	1	1	male	35.0	85	0	0	512.3292	C	890.0
258	1	1	female	35.0	18	0	0	512.3292	C	890.0
679	1	1	male	36.0	1	0	1	512.3292	C	890.0

891 rows × 10 columns

isin

In [28]:

```
embarked = ['S', 'C']
```

In [29]:

```
data[data.Embarked.isin(embarked)]
```

Out[29]:

	Survived	Pclass	Sex	Age	column	SibSp	Parch	Fare	Embarked	Rank
0	0	3	male	22.0	77	1	0	7.2500	S	77.0
1	1	1	female	38.0	4	1	0	71.2833	C	789.0
2	1	3	female	26.0	16	0	0	7.9250	S	232.5
3	1	1	female	35.0	54	1	0	53.1000	S	748.0
4	0	3	male	35.0	3	0	0	8.0500	S	264.0
...	...	...	...	...	...	...	...	...	...	...
884	0	3	male	25.0	58	0	0	7.0500	S	33.0
886	0	2	male	27.0	98	0	0	13.0000	S	407.5
887	1	1	female	19.0	17	0	0	30.0000	S	654.5
888	0	3	female	NaN	16	1	2	23.4500	S	546.5
889	1	1	male	26.0	87	0	0	30.0000	C	654.5

812 rows × 10 columns

replace

In [30]:

```
data.Sex.replace({'male':0, 'female':1}, inplace=True)
data
```

Out[30]:

	Survived	Pclass	Sex	Age	column	SibSp	Parch	Fare	Embarked	Rank
0	0	3	0	22.0	77	1	0	7.2500	S	77.0
1	1	1	1	38.0	4	1	0	71.2833	C	789.0
2	1	3	1	26.0	16	0	0	7.9250	S	232.5
3	1	1	1	35.0	54	1	0	53.1000	S	748.0
4	0	3	0	35.0	3	0	0	8.0500	S	264.0
...	...	...	...	...	...	...	...	...	...	...
886	0	2	0	27.0	98	0	0	13.0000	S	407.5
887	1	1	1	19.0	17	0	0	30.0000	S	654.5
888	0	3	1	NaN	16	1	2	23.4500	S	546.5
889	1	1	0	26.0	87	0	0	30.0000	C	654.5
890	0	3	0	32.0	62	0	0	7.7500	Q	123.5

891 rows × 10 columns

rename

In [31]:

```
data.rename(columns={'Survived':'SURVIVED'}, inplace=True)
```

In [32]:

```
data
```

Out[32]:

	SURVIVED	Pclass	Sex	Age	column	SibSp	Parch	Fare	Embarked	Rank
0	0	3	0	22.0	77	1	0	7.2500	S	77.0
1	1	1	1	38.0	4	1	0	71.2833	C	789.0
2	1	3	1	26.0	16	0	0	7.9250	S	232.5
3	1	1	1	35.0	54	1	0	53.1000	S	748.0
4	0	3	0	35.0	3	0	0	8.0500	S	264.0
...	...	...	...	...	...	...	...	...	...	...
886	0	2	0	27.0	98	0	0	13.0000	S	407.5
887	1	1	1	19.0	17	0	0	30.0000	S	654.5
888	0	3	1	NaN	16	1	2	23.4500	S	546.5
889	1	1	0	26.0	87	0	0	30.0000	C	654.5
890	0	3	0	32.0	62	0	0	7.7500	Q	123.5

891 rows × 10 columns

fillna

In [33]:

```
data.isnull().sum()
```

Out[33]:

```
SURVIVED      0
Pclass        0
Sex           0
Age         177
column        0
SibSp         0
Parch         0
Fare          0
Embarked      2
Rank          0
dtype: int64
```

In [34]:

```
Age_Mean = data.Age.mean()
```

In [35]:

```
Age_Mean
```

Out[35]:

```
29.69911764705882
```

In [36]:

```
data.Age.fillna(Age_Mean, inplace=True)
```

In [37]:

```
data.isnull().sum()
```

Out[37]:

```
SURVIVED    0
Pclass      0
Sex         0
Age         0
column      0
SibSp       0
Parch       0
Fare        0
Embarked    2
Rank        0
dtype: int64
```

pct_change

In [38]:

```
data.Fare.pct_change()
```

Out[38]:

```
0           NaN
1      8.832179
2     -0.888824
3      5.700315
4     -0.848399
         ...
886   -0.553648
887    1.307692
888   -0.218333
889    0.279318
890   -0.741667
Name: Fare, Length: 891, dtype: float64
```

count

In [39]:

```
data.Fare.count()
```

Out[39]:

```
891
```

pd.cut

In [42]:

```
cutoff = [0, 18, 50, 85]
labels = ['Child', 'Adult', 'Old']
data['AGE_TYPE'] = pd.cut(data.Age, bins=cutoff, labels=labels)
```

In [43]:

```
data
```

Out[43]:

	SURVIVED	Pclass	Sex	Age	column	SibSp	Parch	Fare	Embarked	Rank	A
0	0	3	0	22.000000	77	1	0	7.2500	S	77.0	
1	1	1	1	38.000000	4	1	0	71.2833	C	789.0	
2	1	3	1	26.000000	16	0	0	7.9250	S	232.5	
3	1	1	1	35.000000	54	1	0	53.1000	S	748.0	
4	0	3	0	35.000000	3	0	0	8.0500	S	264.0	
...	...	...	...	...	...	...	...	...	...	...	
886	0	2	0	27.000000	98	0	0	13.0000	S	407.5	
887	1	1	1	19.000000	17	0	0	30.0000	S	654.5	
888	0	3	1	29.699118	16	1	2	23.4500	S	546.5	
889	1	1	0	26.000000	87	0	0	30.0000	C	654.5	
890	0	3	0	32.000000	62	0	0	7.7500	Q	123.5	

891 rows × 11 columns

n_largets and n-smallest

In [40]:

```
data.nlargest(5, 'Age')
```

Out[40]:

	SURVIVED	Pclass	Sex	Age	column	SibSp	Parch	Fare	Embarked	Rank
630	1	1	0	80.0	35	0	0	30.0000	S	654.5
851	0	3	0	74.0	43	0	0	7.7750	S	148.5
96	0	1	0	71.0	65	0	0	34.6542	C	692.0
493	0	1	0	71.0	82	0	0	49.5042	C	729.5
116	0	3	0	70.5	17	0	0	7.7500	Q	123.5

In [41]:

```
data.nsmallest(5, 'Age')
```

Out[41]:

	SURVIVED	Pclass	Sex	Age	column	SibSp	Parch	Fare	Embarked	Rank
803	1	3	0	0.42	76	0	1	8.5167	C	294.0
755	1	2	0	0.67	25	1	1	14.5000	S	454.0
469	1	3	1	0.75	10	2	1	19.2583	C	509.5
644	1	3	1	0.75	83	2	1	19.2583	C	509.5
78	1	2	0	0.83	21	0	2	29.0000	S	642.5

CROSS TABULATION

In statistics, a contingency table (also known as a cross tabulation or crosstab) is a type of table in a matrix format that displays the (multivariate) frequency distribution of the variables. They are heavily used in survey research, business intelligence, engineering, and scientific research. They provide a basic picture of the interrelation between two variables and can help find interactions between them. The term contingency table was first used by Karl Pearson in "On the Theory of Contingency and Its Relation to Association and Normal Correlation",[1] part of the Drapers' Company Research Memoirs Biometric Series I published in 1904.

In [44]:

```
pd.crosstab(data.Pclass, data.Embarked)
```

Out[44]:

Embarked	C	Q	S
Pclass			
1	85	2	127
2	17	3	164
3	66	72	353

In [45]:

```
pd.crosstab(data.Pclass, data.Embarked, margins=True)
```

Out[45]:

Embarked	C	Q	S	All
Pclass				
1	85	2	127	214
2	17	3	164	184
3	66	72	353	491
All	168	77	644	889