```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
df = pd.read_csv('train.csv')
```

```
df.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S

Why do EDA

- Model building
- Analysis and reporting
- Validate assumptions
- Handling missing values
- feature engineering
- detecting outliers

```
# Remember it is an iterative process
```

Column Types

- **Numerical** - Age,Fare,PassengerId
- **Categorical** - Survived, Pclass, Sex, SibSp, Parch,Embarked
- **Mixed** - Name, Ticket, Cabin

Univariate Analysis

Univariate analysis focuses on analyzing each feature in the dataset independently.

- **Distribution analysis**: The distribution of each feature is examined to identify its shape, central tendency, and dispersion.
- **Identifying potential issues**: Univariate analysis helps in identifying potential problems with the data such as outliers, skewness, and missing values

The shape of a data distribution refers to its overall pattern or form as it is represented on a graph. Some common shapes of data distributions include:

- **Normal Distribution**: A symmetrical and bell-shaped distribution where the mean, median, and mode are equal and the majority of the data falls in the middle of the distribution with gradually decreasing frequencies towards the tails.
- **Skewed Distribution**: A distribution that is not symmetrical, with one tail being longer than the other. It can be either positively skewed (right-skewed) or negatively skewed (left-skewed).
- **Bimodal Distribution**: A distribution with two peaks or modes.
- **Uniform Distribution**: A distribution where all values have an equal chance of occurring.

The shape of the data distribution is important in identifying the presence of outliers, skewness, and the type of statistical tests and models that can be used for further analysis.

Dispersion is a statistical term used to describe the spread or variability of a set of data. It measures how far the values in a data set are spread out from the central tendency (mean, median, or mode) of the data.

There are several measures of dispersion, including:

- **Range**: The difference between the largest and smallest values in a data set.

- **Variance**: The average of the squared deviations of each value from the mean of the data set.
- **Standard Deviation**: The square root of the variance. It provides a measure of the spread of the data that is in the same units as the original data.
- **Interquartile range (IQR)**: The range between the first quartile (25th percentile) and the third quartile (75th percentile) of the data.

Dispersion helps to describe the spread of the data, which can help to identify the presence of outliers and skewness in the data.

Steps of doing Univariate Analysis on Numerical columns

- **Descriptive Statistics**: Compute basic summary statistics for the column, such as mean, median, mode, standard deviation, range, and quartiles. These statistics give a general understanding of the distribution of the data and can help identify skewness or outliers.
- **Visualizations**: Create visualizations to explore the distribution of the data. Some common visualizations for numerical data include histograms, box plots, and density plots. These visualizations provide a visual representation of the distribution of the data and can help identify skewness an outliers.
- **Identifying Outliers**: Identify and examine any outliers in the data. Outliers can be identified using visualizations. It is important to determine whether the outliers are due to measurement errors, data entry errors, or legitimate differences in the data, and to decide whether to include or exclude them from the analysis.
- **Skewness**: Check for skewness in the data and consider transforming the data or using robust statistical methods that are less sensitive to skewness, if necessary.
- **Conclusion**: Summarize the findings of the EDA and make decisions about how to proceed with further analysis.

Age

conclusions

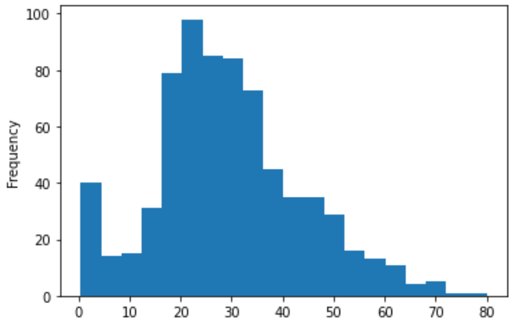
- Age is normally(almost) distributed
- 20% of the values are missing
- There are some outliers

```
df['Age'].describe()
```

```
count    714.000000
mean      29.699118
std       14.526497
min        0.420000
25%       20.125000
50%       28.000000
75%       38.000000
max       80.000000
Name: Age, dtype: float64
```

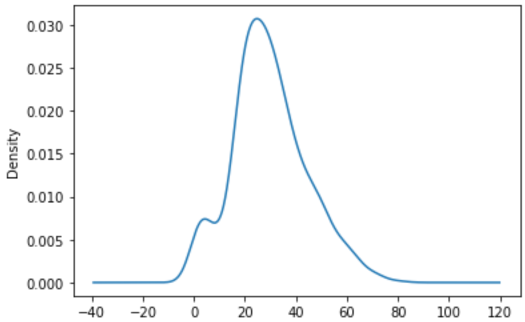
```
df['Age'].plot(kind='hist',bins=20)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f003a0ab1f0>
```




```
df['Age'].plot(kind='kde')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0039fee1f0>
```

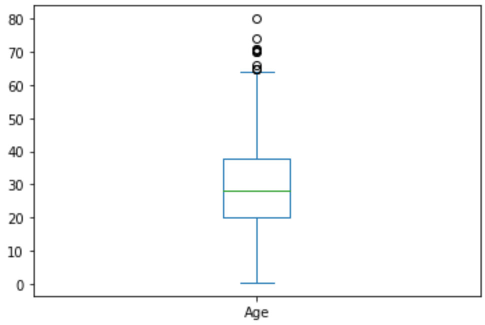



```
df['Age'].skew()
```

```
0.38910778230082704
```

```
df['Age'].plot(kind='box')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0039fd0a60>
```




```
df[df['Age'] > 65]
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
33	34	0	2	Wheadon, Mr. Edward H	male	66.0	0	0	C.A. 24579	10.5000	NaN	S
96	97	0	1	Goldschmidt, Mr. George B	male	71.0	0	0	PC 17754	34.6542	A5	C
116	117	0	3	Connors, Mr. Patrick	male	70.5	0	0	370369	7.7500	NaN	Q
493	494	0	1	Artagaveytia, Mr. Ramon	male	71.0	0	0	PC 17609	49.5042	NaN	C
630	631	1	1	Barkworth, Mr. Algernon Henry Wilson	male	80.0	0	0	27042	30.0000	A23	S
672	673	0	2	Mitchell, Mr. Henry Michael	male	70.0	0	0	C.A. 24580	10.5000	NaN	S
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	WE/P	[illegible]	[illegible]	[illegible]

```
df['Age'].isnull().sum()/len(df['Age'])
```

```
0.19865319865319866
```

Fare

conclusions

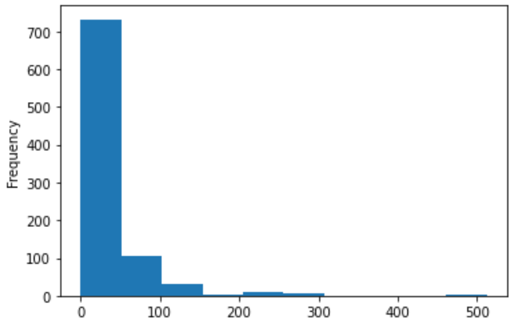
- The data is highly(positively) skewed
- Fare col actually contains the group fare and not the individual fare(This migth be and issue)
- We need to create a new col called individual fare

```
df['Fare'].describe()
```

```
count    891.000000
mean      32.204208
std       49.693429
min        0.000000
25%        7.910400
50%       14.454200
75%       31.000000
max      512.329200
Name: Fare, dtype: float64
```

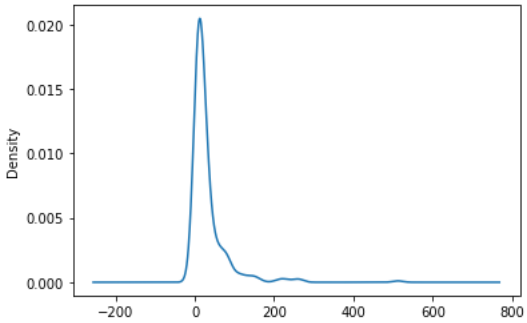
```
df['Fare'].plot(kind='hist')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0037efa910>
```




```
df['Fare'].plot(kind='kde')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0037f48be0>
```

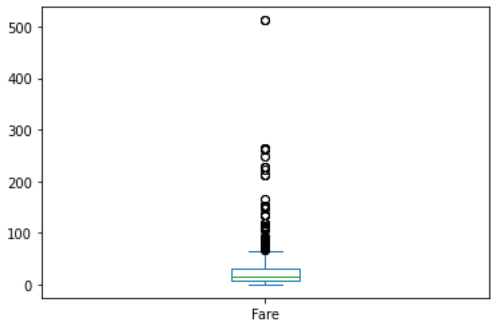



```
df['Fare'].skew()
```

```
4.787316519674893
```

```
df['Fare'].plot(kind='box')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0037dd50a0>
```




```
df[df['Fare'] > 250]
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
27	28	0	1	Fortune, Mr. Charles Alexander	male	19.0	3	2	19950
88	89	1	1	Fortune, Miss. Mabel Helen	female	23.0	3	2	19950
258	259	1	1	Ward, Miss. Anna	female	35.0	0	0	PC 17755
311	312	1	1	Ryerson, Miss. Emily	female	18.0	2	2	PC 17608

```
df['Fare'].isnull().sum()
```

```
0
```

Steps of doing Univariate Analysis on Categorical columns

Descriptive Statistics: Compute the frequency distribution of the categories in the column. This will give a general understanding of the distribution of the categories and their relative frequencies.

Visualizations: Create visualizations to explore the distribution of the categories. Some common visualizations for categorical data include count plots and pie charts. These visualizations provide a visual representation of the distribution of the categories and can help identify any patterns or anomalies in the data.

Missing Values: Check for missing values in the data and decide how to handle them. Missing values can be imputed or excluded from the analysis, depending on the research question and the data set.

Conclusion: Summarize the findings of the EDA and make decisions about how to proceed with further analysis.

Survived

conclusions

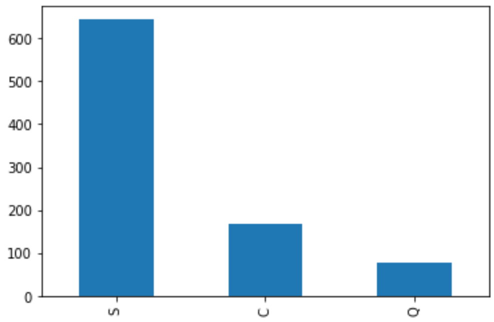
- Parch and SibSp cols can be merged to form a new col call family_size
- Create a new col called is_alone

```
df['Embarked'].value_counts()
```

```
S    644
C    168
Q     77
Name: Embarked, dtype: int64
```

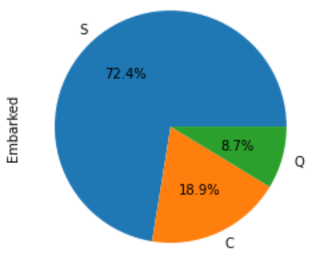
```
df['Embarked'].value_counts().plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0037a68d00>
```




```
df['Embarked'].value_counts().plot(kind='pie',autopct='%0.1f%%')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f0037a9cb50>
```




```
df['Sex'].isnull().sum()
```

```
0
```

Steps of doing Bivariate Analysis

- Select 2 cols
- Understand type of relationship
  1. **Numerical - Numerical**
     a. You can plot graphs like scatterplot(regression plots), 2D histplot, 2D KDEplots
     b. Check correlation coefficent to check linear relationship
  2. **Numerical - Categorical** - create visualizations that compare the distribution of the numerical data across different categories of the categorical data.
     a. You can plot graphs like barplot, boxplot, kdeplot violinplot even scatterplots
  3. **Categorical - Categorical**
     a. You can create cross-tabulations or contingency tables that show the distribution of values in one categorical column, grouped by the values in the other categorical column.
     b. You can plots like heatmap, stacked barplots, treemaps
- Write your conclusions

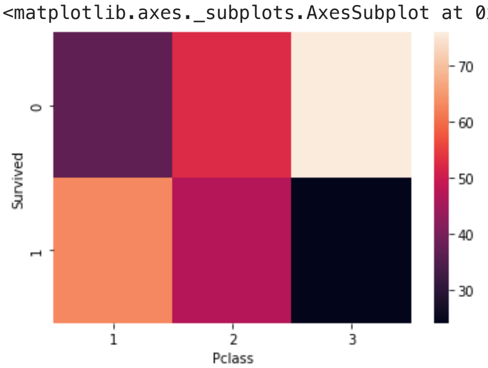
```
df
```

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

Allen, Mr

```
sns.heatmap(pd.crosstab(df['Survived'],df['Pclass'],normalize='columns')*100)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f0037e1adc0>



```
pd.crosstab(df['Survived'],df['Sex'],normalize='columns')*100
```

Sex	female	male
Survived		
0	25.796178	81.109185
1	74.203822	18.890815

```
pd.crosstab(df['Survived'],df['Embarked'],normalize='columns')*100
```

Embarked	C	Q	S
Survived			
0	44.642857	61.038961	66.304348
1	55.357143	38.961039	33.695652

```
pd.crosstab(df['Sex'],df['Embarked'],normalize='columns')*100
```

Embarked	C	Q	S
Sex			
female	43.452381	46.753247	31.521739
male	56.547619	53.246753	68.478261

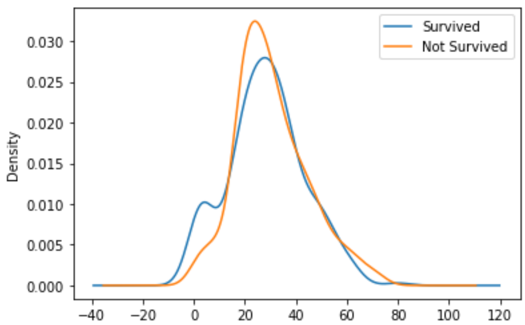
```
pd.crosstab(df['Pclass'],df['Embarked'],normalize='columns')*100
```

Embarked	C	Q	S
Pclass			
1	50.595238	2.597403	19.720497
2	10.119048	3.896104	25.465839
3	39.285714	93.506494	54.813665

```
# survived and age

df[df['Survived'] == 1]['Age'].plot(kind='kde',label='Survived')
df[df['Survived'] == 0]['Age'].plot(kind='kde',label='Not Survived')

plt.legend()
plt.show()
```




```
df[df['Pclass'] == 1]['Age'].mean()
```

```
38.233440860215055
```

```
# Feature Engineering on Fare col
```

```
df['SibSp'].value_counts()
```

```
0    608
1    209
2     28
4     18
3     16
8      7
5      5
Name: SibSp, dtype: int64
```

```
df[df['Ticket'] == 'CA. 2343']
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
159	160	0	3	Sage, Master. Thomas Henry	male	NaN	8	2	CA. 2343	69.55	NaN	S
180	181	0	3	Sage, Miss. Constance Gladys	female	NaN	8	2	CA. 2343	69.55	NaN	S
201	202	0	3	Sage, Mr. Frederick	male	NaN	8	2	CA. 2343	69.55	NaN	S
324	325	0	3	Sage, Mr. George John Jr	male	NaN	8	2	CA. 2343	69.55	NaN	S
792	793	0	3	Sage, Miss. Stella Anna	female	NaN	8	2	CA. 2343	69.55	NaN	S
846	847	0	3	Sage, Mr. Douglas Bullen	male	NaN	8	2	CA. 2343	69.55	NaN	S
863	864	0	3	Sage, Miss. Dorothy Edith "Dolly"	female	NaN	8	2	CA. 2343	69.55	NaN	S

```
df[df['Name'].str.contains('Sage')]
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
159	160	0	3	Sage, Master. Thomas Henry	male	NaN	8	2	CA. 2343	69.55	NaN	S
180	181	0	3	Sage, Miss. Constance Gladys	female	NaN	8	2	CA. 2343	69.55	NaN	S
201	202	0	3	Sage, Mr. Frederick	male	NaN	8	2	CA. 2343	69.55	NaN	S
324	325	0	3	Sage, Mr. George John Jr	male	NaN	8	2	CA. 2343	69.55	NaN	S
641	642	1	1	Sagesser, Mlle. Emma	female	24.0	0	0	PC 17477	69.30	B35	C
792	793	0	3	Sage, Miss. Stella Anna	female	NaN	8	2	CA. 2343	69.55	NaN	S
846	847	0	3	Sage, Mr. Douglas Bullen	male	NaN	8	2	CA. 2343	69.55	NaN	S
863	864	0	3	Sage, Miss. Dorothy Edith "Dolly"	female	NaN	8	2	CA. 2343	69.55	NaN	S

```
df1 = pd.read_csv('/content/test.csv')
```

```
df = pd.concat([df,df1])
```

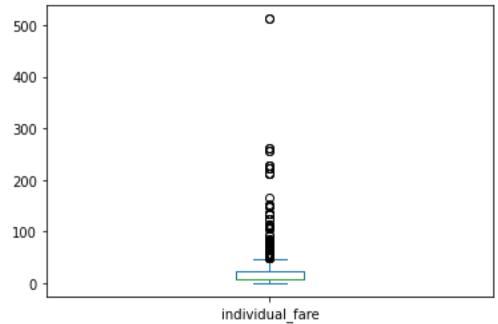
```
df[df['Ticket'] == 'CA 2144']
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
59	60	0.0	3	Goodwin, Master. William Frederick	male	11.0	5	2	CA 2144	46.9	NaN	S
71	72	0.0	3	Goodwin, Miss. Lillian Amy	female	16.0	5	2	CA 2144	46.9	NaN	S
386	387	0.0	3	Goodwin, Master. Sidney Leonard	male	1.0	5	2	CA 2144	46.9	NaN	S
480	481	0.0	3	Goodwin, Master. Harold Victor	male	9.0	5	2	CA 2144	46.9	NaN	S
678	679	0.0	3	Goodwin, Mrs. Frederick (Augusta Tyler)	female	43.0	1	6	CA 2144	46.9	NaN	S
									CA			

```
df['individual_fare'] = df['Fare']/(df['SibSp'] + df['Parch'] + 1)
```

```
df['individual_fare'].plot(kind='box')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f00347f4c40>
```




```
df[['individual_fare','Fare']].describe()
```

	individual_fare	Fare
count	1308.000000	1308.000000
mean	20.518215	33.295479
std	35.774337	51.758668
min	0.000000	0.000000
25%	7.452767	7.895800
50%	8.512483	14.454200
75%	24.237500	31.275000
max	512.329200	512.329200

```
df['Fare'].
```

```
0        7.2500
1       71.2833
2        7.9250
3       53.1000
4        8.0500
         ...
413      8.0500
414    108.9000
415      7.2500
416      8.0500
417     22.3583
Name: Fare, Length: 1309, dtype: float64
```

```
df
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Tick
0	1	0.0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 211
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 175
2	3	1.0	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/(31012
3	4	1.0	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	1138
4	5	0.0	3	Allen, Mr. William Henry	male	35.0	0	0	3734

```
df['family_size'] = df['SibSp'] + df['Parch'] + 1
```

```
# family_type
# 1 -> alone
# 2-4 -> small
# >5 -> large

def transform_family_size(num):

  if num == 1:
    return 'alone'
  elif num>1 and num <5:
    return "small"
  else:
    return "large"
```

```
df['family_type'] = df['family_size'].apply(transform_family_size)
```

```
df
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Tick
0	1	0.0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 211
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 175
2	3	1.0	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/(31012
3	4	1.0	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	1138
4	5	0.0	3	Allen, Mr. William Henry	male	35.0	0	0	3734
...	...	...	...	...	...	...	...	...	
413	1305	NaN	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 32
414	1306	NaN	1	Oliva y Ocana,	female	39.0	0	0	PC 177

```
pd.crosstab(df['Survived'],df['family_type'],normalize='columns')*100
```

family_type	alone	large	small
Survived			
0.0	69.646182	83.870968	42.123288
1.0	30.353818	16.129032	57.876712

```
df['surname'] = df['Name'].str.split(',').str.get(0)
```

```
df
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Tick
0	1	0.0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 211
1	2	1.0	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 175
2	3	1.0	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/(31012
3	4	1.0	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	1138
4	5	0.0	3	Allen, Mr. William Henry	male	35.0	0	0	3734
...	...	...	...	...	...	...	...	...	
413	1305	NaN	3	Spector, Mr. Woolf	male	NaN	0	0	A.5. 32
414	1306	NaN	1	Oliva y Ocana, Dona. Fermina	female	39.0	0	0	PC 177
415	1307	NaN	3	Saether, Mr. Simon Sivertsen	male	38.5	0	0	SOTON/O 31012
416	1308	NaN	3	Ware, Mr. Frederick	male	NaN	0	0	3593
417	1309	NaN	3	Peter, Master. Michael J	male	NaN	1	1	26

1309 rows × 16 columns

```
df['title'] = df['Name'].str.split(',').str.get(1).str.strip().str.split(' ').str.get(0)
```

```
temp_df = df[df['title'].isin(['Mr.','Miss.','Mrs.','Master.','ootherr'])]
```

```
pd.crosstab(temp_df['Survived'],temp_df['title'],normalize='columns')*100
```

title	Master.	Miss.	Mr.	Mrs.	ootherr
Survived					
0.0	42.5	30.21978	84.332689	20.8	72.222222
1.0	57.5	69.78022	15.667311	79.2	27.777778

```
df['title'] = df['title'].str.replace('Rev.','other')
df['title'] = df['title'].str.replace('Dr.','other')
df['title'] = df['title'].str.replace('Col.','other')
df['title'] = df['title'].str.replace('Major.','other')
df['title'] = df['title'].str.replace('Capt.','other')
df['title'] = df['title'].str.replace('the','other')
df['title'] = df['title'].str.replace('Jonkheer.','other')
# ,'Dr.','Col.','Major.','Don.','Capt.','the','Jonkheer.']
```

```
<ipython-input-124-ffb23deeff2c>:1: FutureWarning: The default value of regex will change from True to False in a future
  df['title'] = df['title'].str.replace('Rev.','other')
<ipython-input-124-ffb23deeff2c>:2: FutureWarning: The default value of regex will change from True to False in a future
  df['title'] = df['title'].str.replace('Dr.','other')
<ipython-input-124-ffb23deeff2c>:3: FutureWarning: The default value of regex will change from True to False in a future
  df['title'] = df['title'].str.replace('Col.','other')
<ipython-input-124-ffb23deeff2c>:4: FutureWarning: The default value of regex will change from True to False in a future
  df['title'] = df['title'].str.replace('Major.','other')
<ipython-input-124-ffb23deeff2c>:5: FutureWarning: The default value of regex will change from True to False in a future
```

```
  df['title'] = df['title'].str.replace('Capt.','other')
<ipython-input-124-ffb23deeff2c>:7: FutureWarning: The default value of regex will change from True to False in a future
  df['title'] = df['title'].str.replace('Jonkheer.','other')
```

```
df['Cabin'].isnull().sum()/len(df['Cabin'])
```

```
0.774637127578304
```

```
df['Cabin'].fillna('M',inplace=True)
```

```
df['Cabin'].value_counts()
```

```
M                  1014
C23 C25 C27           6
B57 B59 B63 B66       5
G6                    5
F33                   4
                   ...
A14                   1
E63                   1
E12                   1
E38                   1
C105                  1
Name: Cabin, Length: 187, dtype: int64
```

```
df['deck'] = df['Cabin'].str[0]
```

```
df['deck'].value_counts()
```

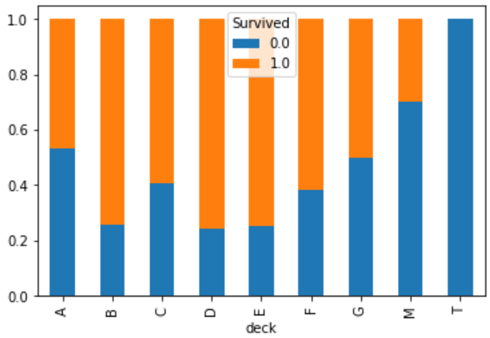
```
M    1014
C      94
B      65
D      46
E      41
A      22
F      21
G       5
T       1
Name: deck, dtype: int64
```

```
pd.crosstab(df['deck'],df['Pclass'])
```

Pclass	1	2	3
deck			
A	22	0	0
B	65	0	0
C	94	0	0
D	40	6	0
E	34	4	3
F	0	13	8
G	0	0	5
M	67	254	693
T	1	0	0

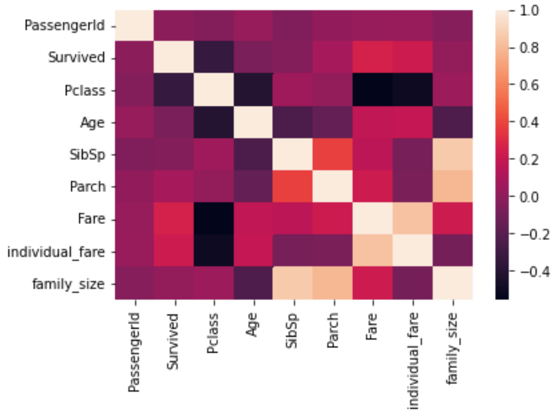
```
pd.crosstab(df['deck'],df['Survived'],normalize='index').plot(kind='bar',stacked=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f00343b9af0>
```



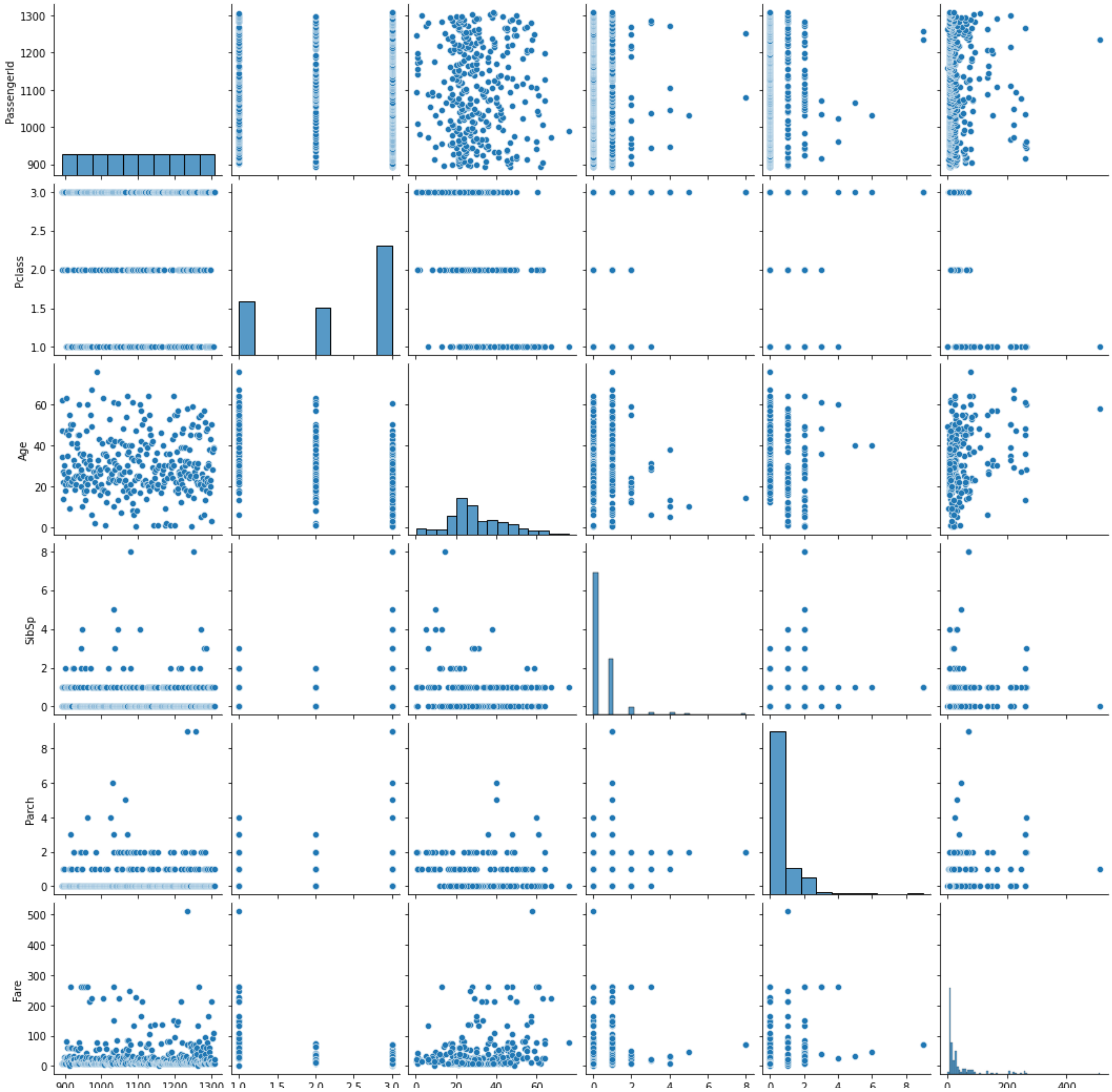
```
sns.heatmap(df.corr())
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f00343513d0>
```




```
sns.pairplot(df1)
```

```
<seaborn.axisgrid.PairGrid at 0x7f003427fdc0>
```




```
df1
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Far
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.829
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.000
			Myles,						